VMS Glossary

Order Number: AA–LA03B–TE

November 1991

This document contains definitions for commonly used terms in documentation of the VMS operating system.

Revision/Update Information: This revised document supersedes the *VMS Glossary*, Version 5.0.

Software Version: VMS Version 5.5

Digital Equipment Corporation
Maynard, Massachusetts

November 1991

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

ZK4507

This document was prepared using VAX DOCUMENT, Version 1.2

Preface

Intended Audience

This document is intended for all users of VMS documentation.

Document Structure

The VMS terms are presented in alphabetical order. They have not been grouped into categories.

Glossary

$allocation-class$

Part of a naming convention for physical devices. This convention is described in the *VMS VAXcluster Manual*.

"a" character

The subset of ASCII characters that are allowed in the labels of ANSI-labeled magnetic tape.

abort

An exception occurring in the middle of an instruction that sometimes leaves the registers and memory in an indeterminate state such that the instruction cannot necessarily be restarted.

absolute indexed mode

An indexed addressing mode in which the base operand specifier is addressed in absolute mode. See also *indexed addressing mode* and *absolute mode*.

absolute mode

A mode of address in which the program counter (PC) is used as the register in autoincrement deferred mode. The PC contains the address of the location containing the actual operand.

absolute time

A value expressing a specific date (month, day, and year) and time of day. Absolute time values are always expressed in the system as positive numbers.

access control

Validating requests for connection, login, or file access to determine whether or not they can be accepted. User name and password provide the most common means of access control.

access control list (ACL)

A list that defines the kinds of access to be granted or denied to users of an object. Access control lists can be created for objects such as files, devices, and mailboxes. Each access control list consists of one or more entries known as access control list entries.

access control list entry (ACE)

An entry in an access control list. Access control list entries may specify identifiers and the access rights to be granted or denied the holders of the identifiers, default protection for directories, or security alarm details. Access control lists for each object can hold numerous entries, limited only by overall space and performance considerations.

access mode

A processing mode that determines which instructions a processor can execute and which locations in memory it can read and write. The four processor access modes are, in order from most to least privileged and protected, kernel (mode 0), executive (mode 1), supervisor (mode 2), and user (mode 3). When the processor is in kernel mode, the executing software has complete control of, and responsibility for, the system. The processor status longword contains the current access mode field. The operating system uses access modes to define protection levels for software executing in the context of a process. For example, the executive and most system services run in kernel mode and are most protected. VMS Record Management Services (RMS) and some system services run in executive mode. The command interpreter is less protected and runs in supervisor mode. The debugger runs in user mode with the same privileges and protection as normal user programs. See also *record access mode*.

access type

(1) The way the processor interprets instruction operands. Access types are read, write, modify, address, and branch.

(2) The way a procedure accesses its arguments. See also *record access type*.

access violation

An exception that takes place on an attempt to reference an address that is either not mapped into virtual memory or not accessible by the current access mode.

account

A character-string name or number that identifies an individual user when the user logs in. That name or number tells the system where the user's files are and the type of access to other files the user has.

User accounts are either privileged or nonprivileged. Privileged users have access to privileged system commands. Nonprivileged users are limited to everyday operations that do not threaten the integrity of the operating system.

accounting manager

A function in the job controller system process that writes accounting records to a system accounting log file to track job activity for user process termination, printer and batch jobs, and so on.

account name

A string that identifies a particular account used to accumulate data on a job's resource use. This name is the user's accounting charge number, not the user identification code (UIC).

ACE

See *access control list entry*.

ACL

See *access control list*.

ACP

See *ancillary control process*.

active component

A network component whose operational state is other than OFF. The ACTIVE keyword can be used with the Network Control Program (NCP) commands SHOW or LIST to display information about active lines, circuits, nodes, and logging.

active member

A shadow set member that is fully consistent with all other active members of the shadow set.

active set

In a VMS symmetric multiprocessing system, those processors that have been bootstrapped into the system, have undergone initialization, and are capable of scheduling and executing processes. Together, the primary processor and all secondary processors make up a system's active set. Compare with *available set*.

adapter control block (ADP)

A structure in the I/O database that describes an adapter or device. VMS creates an ADP for each UNIBUS or MASSBUS adapter, each UNIBUS or MASSBUS device, and each device on a VAXBI bus.

address

A number used by the operating system and user software to identify a storage location in memory. See also *virtual address* and *physical address*.

address access type

A type of access in which the specified operand of an instruction is not accessed directly by the instruction. The address of the specified operand is the actual instruction operand, and the context of the address calculation is determined by the data type of the operand.

address space

The set of all possible addresses available to a process. Virtual address space refers to the set of all possible virtual addresses. Physical address space refers to the set of all possible physical addresses transmitted on the backplane interconnect.

addressing mode

The way in which an operand is specified; for example, the way in which the effective address of an instruction operand is calculated using the general registers. The seven general-register addressing modes are register, register deferred, autoincrement, autoincrement deferred, autodecrement, displacement, and displacement deferred. In addition, there are six indexed addressing modes using two general registers and literal mode addressing. The three program counter (PC) addressing modes are referred to as immediate (for register deferred mode using the PC), absolute (for autoincrement deferred mode using the PC), and branch.

adjacent node

A network node connected to the local node by a single physical line.

ADP

See *adapter control block*.

affinity

In a VMS symmetric multiprocessing system, a close association of a device or a process with a specific processor or set of processors in the system. See *device affinity* and *process affinity*.

alarm

See *security alarm*.

alert

A mechanism by which a thread informs either itself or another thread to terminate as soon as possible. *Alert* and *cancel* have the same meaning.

alertable

A routine in which synchronous alert delivery can occur only at specific, well-defined points. These points are DECthreads routines that determine if an alert is pending and, if so, deliver the alert.

alertsafe

A routine that can be called without risk of triggering an alert while asynchronous delivery of alerts is enabled.

allocating

Reserving a particular device unit for exclusive use. A user process can allocate a device only when that device is not allocated by any other process.

allocation class

A unique number between 0 and 255 that the system manager assigns to a pair of hosts and to the dual-pathed devices that the hosts make available to other nodes in the VAXcluster configuration. (Allocation class 0 is used to denote that the node is not a member of any allocation class.) The allocation class provides a way for users to access dual-pathed devices through either of the hosts. In this way, if one host of an allocation class set is not available, the user can gain access to a device specified by that allocation class through the other host of the allocation class.

alphanumeric character

An uppercase or lowercase letter, a dollar sign, an underscore, or a decimal digit.

alphanumeric UIC

A format of user identification code (UIC) that specifies the user's group (optional) and member in alphanumeric form rather than numeric form.

alternate key

An optional key within the data records in an indexed file; used by VMS Record Management Services (RMS) to build an alternate index. See *key (indexed files)* and *primary key*.

American Standard Code for Information Interchange (ASCII)

A set of 8-bit binary numbers representing the alphabet, punctuation, numerals, and other special symbols used in text representation and communications protocol.

ancillary control process (ACP)

A process that acts as an interface between user software and an I/O driver. An ACP provides functions supplemental to those performed in the driver, such as file and directory management. Two examples of ACPs are the magnetic tape ACP and the network ACP.

AP

See *argument pointer*.

application

A set of tasks related by the business activity they support and controlled as a single unit. See also *transaction processing*.

area

(1) A region of an indexed file maintained by VMS Record Management Services (RMS) that allows a user to specify placement or specific bucket sizes or both for particular portions of a file. An area consists of any number of buckets; there may be from 1 to 255 areas in a file.

(2) A group of nodes in a network that can run independently as a subnetwork.

area router

See *level 2 router*.

area routing

A technique for grouping the nodes in a network into areas for routing purposes. Routing in a multiple-area network is on two levels, with one level of routing within an area (called level 1 routing) and a second, higher level of routing between areas (called level 2 routing). Area routing permits a DECnet network to support the configuration of very large networks.

argument pointer (AP)

General register 12 (R12). By convention, AP contains the address of the base of the argument list for procedures initiated using CALL instructions.

ASCII

See *American Standard Code for Information Interchange*.

ASMP

See *asymmetric multiprocessing*.

asymmetric multiprocessing (ASMP)

A multiprocessing configuration in which the processors are not equal in their ability to execute operating system code. In general, a single processor is designated as the primary, or master, processor; other processors are the slaves. The slave processors are limited to performing certain tasks, whereas the master processor can perform all system tasks. Contrast with *symmetric multiprocessing*.

assembler

A language processor that translates a source program containing assembly language directives and machine instructions into an object module.

assembly language

A machine-oriented programming language. VAX MACRO is the assembly language for the VAX computer. See *binary machine code*.

assigning a channel

Establishing the necessary software link between a user process and a device unit that allows a user process to communicate with the device.

assignment statement

In DCL, the association of a symbol name to use with a character string or numeric value. Symbols can define synonyms for system commands or can be used for variables in command procedures.

AST

See *asynchronous system trap*.

ASTLVL

See *asynchronous system trap level*.

asynchronous record operation

A mode of record processing in which a user program can continue to execute after issuing a record retrieval or storage request without having to wait for the request to be fulfilled.

asynchronous system trap (AST)

A software-simulated interrupt to a user-defined service routine. ASTs enable a user process to be notified asynchronously, with respect to that process, of the occurrence of a specific event. If a user process has defined an AST routine for an event, the system interrupts the process and executes the AST routine when that event occurs. When the AST routine exits, the system resumes execution of the process at the point where it was interrupted.

asynchronous system trap level (ASTLVL)

A value kept in an internal processor register that is the most privileged access mode for which an AST is pending. The AST does not occur until the current access mode drops in privilege (rises in numeric value) to a value greater than or equal to ASTLVL. Thus, an AST for an access mode is not serviced while the processor is executing in a more privileged access mode.

atomicity

A requirement that either all operations of a transaction are made permanent or none of them are made permanent.

attached processor

See *secondary processor*.

attribute

(1) In the security context, an attribute is a field of information maintained in the rights database that identifies a characteristic that may be accorded to a holder of the identifier. For example, if the holder and the identifier possess the resource attribute, the holder of that identifier can charge resources, such as disk quota, to that identifier.

(2) In threads, the individual components of the attributes object. Attributes specify detailed properties about the objects to be created.

attributes object

An object that holds the individual attribute values to be used when creating threads, mutexes, or condition variables. An attributes object is analogous to a type definition in a programming language. It describes details of the objects to be created.

auditing

See *security auditing*.

authentication

The act of establishing the identity of users when they start to use the system. VMS (and most other commercial operating systems) uses passwords as the primary authentication mechanism.

authorization file

See *user authorization file*.

autodecrement indexed mode

An indexed addressing mode in which the base operand specifier uses autodecrement mode addressing. See also *indexed addressing mode* and *autodecrement mode*.

autodecrement mode

An addressing mode in which the contents of the selected register are decremented, and the result is used as the address of the actual operand for the instruction. The contents of the register are decremented according to the size of the operand. See also *addressing mode*.

autoincrement deferred indexed mode

An indexed addressing mode in which the base operand specifier uses autoincrement deferred mode addressing. See also *indexed addressing mode* and *autoincrement deferred mode*.

autoincrement deferred mode

An addressing mode in which the specified register contains the address of a longword that contains the address of the actual operand. The contents of the register are incremented by 4 (the number of bytes in a longword). If the program counter (PC) is used as the register, this mode is called absolute mode. See also *addressing mode*.

autoincrement indexed mode

An indexed addressing mode in which the base operand specifier uses autoincrement mode addressing. See also *indexed addressing mode* and *autoincrement mode*.

autoincrement mode

An addressing mode in which the contents of the specified register are used as the address of the operand; the contents of the register are then incremented by the size of the operand: 1 for byte, 2 for word, 4 for longword and floating, 8 for quadword and double floating. See also *addressing mode*.

automatic record locking

A VMS Record Management Services (RMS) capability that allows a user to lock only one record in a specific shared file at any given time. The lock occurs on every execution of a $FIND or $GET macroinstruction (unless the NLK bit is set in the record processing field). The lock is released when the next record is accessed, when the current record is updated or deleted, when the record stream is disconnected, or when the file is closed. VMS Record Management Services (RMS) maintains locks on modified records during a recovery unit.

available set

In a VMS symmetric multiprocessing system, those processors that have passed the system's power-on hardware diagnostics and may or may not be actively involved in the system. The available set includes the active set. Compare with *active set*.

availability

The percentage or amount of scheduled time that a computing system provides application service.

backplane interconnect

An internal processor bus that allows I/O device controllers to communicate with main memory and the central processor. These I/O controllers may reside on the same bus as memory and the central processor (for instance, in VAX 8200/8250/8300/8350 systems), or they may be on a separate bus entirely (for instance, in a VAX–11/780 or VAX 8600/8650 system). In the latter case, an I/O adapter enables and controls the communications between the I/O bus and the processor and memory.

The backplane interconnect is called the synchronous backplane interconnect (SBI) in the VAX–11/780 and VAX 8600 family of systems, the CPU-to-memory interconnect (CMI) on the VAX–11/750 processor, and the VAXBI in the VAX 8530/8550/8700/8800, VAX 8200/8250/8300/8350, and VAX 6200/6300 families of systems. The MicroVAX 3400/3600/3900 series and MicroVAX II processors use the Q-22 bus as a backplane.

backup switching

When a second computing system or component picks up processing in the event that the primary computing system or component fails.

balance set

The collection of all process working sets currently resident in physical memory. The balance set is maintained by the system's swapper process.

bandwidth

The range of frequencies assigned to a channel or system; that is, the difference expressed in Hertz between the highest and lowest frequencies of a band.

base operand address

The address of the base of a table or array referenced by index mode addressing.

base operand specifier

The register used to calculate the base operand address of a table or array referenced by index mode addressing.

base priority

The priority that the system assigns to a process when it is created. A base priority generally comes from the authorization file. The scheduler never schedules a process below its base priority. The base priority can be modified only by the system manager or the process itself. The base priority of a running process can be altered by any user with ALTPRI privilege.

base register

A general register used to contain the address of the first entry in a list, table, array, or other data structure.

batch job

A noninteractive process.

batch processing

A mode of processing in which all commands to be executed by the operating system (and, optionally, data to be used as input to the commands) are placed in a file or punched onto cards and submitted to the system for execution as a single unit. Compare with *command procedure*.

BCUG

See *bilateral closed user group*.

bilateral closed user group (BCUG)

An optional packet switching data network facility that restricts a pair of DTEs (data terminal equipment) to communicating with each other.

binary machine code

The internal instruction format used by the computer. It is called binary because only two characters—0 and 1—are used in this code. Programmers use languages, compilers, and the assembler to generate the binary code.

binding

See *linking*.

bit string

See *variable-length bit field*.

bits per inch (bpi or bits/in)

The recording density of a magnetic tape; indicating how much data can fit on 1 inch of the recording surface. See *density*.

block

(1) The smallest logically addressable unit of data that a specified device can transfer in an I/O operation (512 contiguous bytes for most disk devices).

(2) An arbitrary number of contiguous bytes used to store logically related status, control, or other processing information.

(3) To prevent a thread from executing.

block I/O

A data-accessing technique in which the program manipulates the blocks (physical records) that make up a file, instead of its logical records; allows for the direct access to the blocks in a file without regard for the file organization or record format.

blocking AST

An AST that can be requested by a process using the lock management system services. A blocking AST is delivered to the requesting process when it is preventing another process from accessing a resource.

boot

Short for bootstrap. A bootstrap is a technique or device by which the system brings itself into a desired state by means of its own action, such as a routine whose first few instructions are sufficient to bring the rest of the routine into memory from an input device. Bringing a fresh operating system into memory is called booting.

boot name

The abbreviated name of the device being used to boot software. For example, DU0 is the boot name for device DU0.

boot server

The management center for a VAXcluster system and its major source provider. The boot server provides disk access and downline loads satellite nodes.

bootstrap block

A block in the index file of a system disk. It can contain a program that loads the operating system into memory.

bpi or bpi/in

See *bits per inch*.

branch access type

An instruction attribute that indicates the processor does not reference an operand address; rather, the operand is a branch displacement. The size of the branch displacement is given by the data type of the operand.

branch mode

An addressing mode in which the instruction operand specifier is a signed byte or word displacement. The displacement is added to the contents of the updated program counter (PC), which is the address of the first byte beyond the displacement, and the result is the branch address.

breach

A break in the system security that results in admitting a person or program to an object.

break-in attempt

An effort made by an unauthorized source to gain access to the system. Since the first system access is achieved through logging in, break-in attempts primarily refer to attempts to log in illegally. These attempts focus on supplying passwords for users known to have accounts on the system through informed guesses or other trial-and-error methods.

broadcast

In threads, a mechanism to wake all threads waiting on a condition variable. See also *signal*.

broadcast addressing

In an Ethernet network, a special type of multicast addressing in which all nodes are to receive a message.

broadcast circuit

A circuit having multiple nodes connected to it and having a method for transmitting a message that will be received by multiple receivers.

bucket

A storage structure of 1 to 32 blocks used for building and processing files of relative and indexed organization. A bucket contains one or more records or record cells. Buckets are the unit of contiguous transfer between VMS Record Management Services (RMS) buffers and the disk.

bucket locking

A facility that prevents access to any record in a bucket by more than one user until that user releases the bucket.

bucket split

The result of inserting records into a full bucket. To minimize bucket splits, VMS Record Management Services (RMS) attempts to keep half of the records in the original bucket and transfers the remaining records to a newly created bucket.

buffer

An internal memory area used for temporary storage of data records during input or output operations.

buffered data path

A UNIBUS adapter data path that transfers 32 or 64 bits of data in a single synchronous backplane interconnect (SBI) transfer.

buffered I/O

An I/O operation, such as terminal or mailbox I/O, in which an intermediate buffer from the system buffer pool is used instead of a process-specified buffer. Contrast with *direct I/O*.

bugcheck

An internal inconsistency that VMS cannot resolve. If a nonfatal bugcheck is declared, an error log entry is made. If the system cannot continue to run, a fatal bugcheck is declared and the system produces a crash dump.

busywait

See *spin wait*.

byte

Eight contiguous bits starting on any addressable boundary. Bits are numbered 0 to 7 from right to left. Bit 0 is the low-order bit. When interpreted arithmetically, a byte is a two's complement integer with significance increasing from bits 0 through 6. Bit 7 is the sign bit. The value of the signed integer is in the range –128 to +127 decimal. When interpreted as an unsigned integer, significance increases from bits 0 through 7 and the value of the unsigned integer is in the range 0 to 255 decimal. A byte can be used to store one ASCII character.

cache memory

A small, high-speed memory placed between slower main memory and the processor. A cache increases effective memory transfer rates and processor speed. It contains copies of data recently used by the processor and fetches several bytes of data from memory in anticipation that the processor will access the next sequential series of bytes.

call

To transfer control to a specified routine.

call frame

See *stack frame*.

call instructions

The processor instructions CALLG (call procedure with general argument list) and CALLS (call procedure with stack argument list).

call stack

The stack and the conventional stack structure used during a procedure call. Each access mode of each process context has one call stack, and the interrupt service context has one call stack.

cancel

A mechanism by which a thread informs either itself or another thread to terminate as soon as possible. *Cancel* and *alert* have the same meaning.

capability

In a VMS symmetric multiprocessing environment, an attribute of a single processor or set of processors. The capabilities required by a given process determine the set of processors on which it can be scheduled. For instance, the VMS routine that maintains the system time can execute only on the processor that has the timekeeper capability.

captive account

A type of VMS account that limits the activities of the user. Typically, the user is restricted to using certain command procedures and commands. For example, the user may not be allowed to use the Ctrl/Y key. This type of account is synonymous with a turnkey or tied account.

carrier sense

A signal provided by the Physical layer of the DECnet architecture to indicate that one or more stations (nodes) are currently transmitting on the Ethernet channel.

Carrier Sense, Multiple Access with Collision Detect (CSMA/CD)

A link management procedure used by the Ethernet. Allows multiple stations to access the broadcast channel at will, avoids contention by means of carrier sense and deference, and resolves contention by means of collision detection and retransmission.

CCB

See *channel control block*.

CCITT

Comité Consultatif International Télégraphique et Téléphonique (International Telegraph and Telephone Consultative Committee). An international consultative committee that sets international communications usage standards.

central processing unit (CPU)

The hardware that handles all calculating and routing of input and output as well as executing programs. In short, the CPU is the part of the computer that actually computes.

channel

(1) A logical path connecting a user process to a physical device unit. A user process requests the operating system to assign a channel to a device so the process can communicate with that device. Contrast with *controller data channel*.

(2) A means of transmission over a packet switching data network. For the VAX Packetnet System Interface (PSI), a logical path between a DTE (data terminal equipment) and a DCE (data circuit-terminating equipment) over which data is transmitted. Each channel is identified by a unique reference number called a logical channel number (LCN).

channel control block (CCB)

A structure in the I/O database created by the Assign I/O Channel system service to describe the device unit to which a channel is assigned.

channel request block (CRB)

A structure in the I/O database that describes the activity on a particular controller. The channel request block for a controller contains pointers to the wait queue of drivers ready to access a device through the controller.

character

A symbol represented by an ASCII code. See also *alphanumeric character*.

character buffer

A temporary storage area used to store the last character deleted by an EDT delete character operation.

character string

A contiguous set of bytes. A character string is identified by two attributes: an address and a length. Its address is the address of the byte containing the first character of the string. Subsequent characters are stored in bytes of increasing addresses. The length is the number of characters in the string.

character string descriptor

A quadword data structure used for passing character data (strings). The first word of the quadword contains the length of the character string. The second word can contain type information. The remaining longword contains the address of the string.

characteristics

A display type for the Network Control Program (NCP) commands SHOW and LIST. It refers to static information about a component that is kept in either the volatile or permanent database. Such information may include parameters defined for that component by either the SET or DEFINE command.

CI

See *computer interconnect*.

CI-only VAXcluster configuration

A type of VAXcluster configuration in which the computer interconnect (CI) device is used for most interprocessor communication. In these configurations, a cluster node may be a VAX processor or a hierarchical storage controller (HSC).

circuit

Virtual communication paths between nodes or data terminal equipment (DTE). Circuits operate over physical lines and are the media on which all I/O occurs. X.25 circuits are virtual circuits.

client

A computing system entity that uses the services of other system entities called *servers*.

client/server

A style of computing in which a server system provides common database access, performs computations, and assumes system management tasks for its clients.

closed user group (CUG)

An optional packet switching data network facility that restricts two or more DTEs (data terminal equipment) in the same group to communicating with each other. The basic CUG also prevents these DTEs from accessing or being accessed by other DTEs outside the group. Additions to the basic CUG facility allow one or more DTEs to access or be accessed by DTEs outside the group.

cluster

(1) A set of contiguous blocks that is the basic unit of space allocation on a Files–11 disk volume.

(2) A set of pages brought into memory in one paging operation (page fault cluster).

(3) An event flag cluster.

(4) A configuration of VAX processors (VAXcluster configuration). See also *VAXcluster configuration*.

CMI

See *computer memory interconnect*.

CMP

See *compatibility mode bit*.

collating sequence

An order assigned to the characters of a character set (for example, ASCII, multinational, and EBCDIC) used for sequencing purposes.

collision

Multiple network transmissions overlapping in the physical channel, resulting in garbled data and necessitating retransmission.

collision detect

A signal provided by the Physical layer of the DECnet architecture to its Data Link layer to indicate that one or more stations (nodes) are contending with the local station's transmission.

command

In DIGITAL Command Language (DCL), an instruction, generally an English word, entered by the user at a terminal or included in a command procedure. A command requests the software monitoring a terminal or reading a command procedure to perform some well-defined activity. For example, entering the COPY command requests the system to copy the contents of one file into another file.

command file

See *command procedure*.

command interpreter

A procedure-based system code that executes in supervisor mode in the context of a process to receive, to check the syntax of, and to parse commands entered by the user at a terminal or submitted in a command file.

command language interpreter

See *command interpreter*.

command level

Input stream for the command interpreter. The initial input stream is always command level 0. An interactive command procedure begins executing at command level 1. A batch job command procedure begins executing at command level 0. You can use the execute procedure (@) command or the CALL command in a command procedure to create up to 32 nested command levels.

command node

The node from which a Network Control Program (NCP) command is issued.

command parameter

The positional operand of a command delimited by spaces, such as a file specification, option, or constant.

command procedure

A file containing commands and data that the command interpreter can accept in lieu of the user's entering the commands individually on a terminal. Thus, command procedures provide a means of automatically passing commands to the operating system. In addition, they permit users to employ such programming techniques as loops, counters, labels, and symbol substitution to set up elaborate command sequences that can be altered through user interaction. Command procedures can also be submitted to the system for processing as batch jobs.

command string

A line (or set of continued lines) containing a command and, optionally, information modifying the command. A complete command string consists of a command, its qualifiers, if any, and its parameters (file specifications, for example), if any, and their qualifiers, if any. A command string is normally terminated by pressing the Return key.

common

A FORTRAN term for a program section that contains only data.

common event flag cluster

A set of 32 event flags that enables cooperating processes to post event notification to each other. Common event flag clusters are created as they are needed. A process can associate with up to two common event flag clusters.

compatibility mode

A mode of execution that enables the central processor to execute nonprivileged PDP–11 instructions. The operating system supports compatibility mode execution by providing an RSX–11M execution environment for an RSX–11M task image. The operating system compatibility mode procedures intercept calls to the RSX–11M executive and convert them to the appropriate operating system functions. Note that a layered product (VAX–11 RSX) is required for the RSX–11M environment.

compatibility mode bit (CMP)

The compatibility mode bit in the hardware processor status longword.

compiler

A system component that translates a program written in a high-level language into an object module in binary machine code.

complete crash dump

A crash dump that contains the complete contents of all of physical memory.

component

An element in a network that can be controlled and monitored. Components include lines, circuits, nodes, modules, logging, and objects. Components form part of the Network Control Program (NCP) command syntax.

computer interconnect (CI)

A high-speed, fault-tolerant, dual-path bus, which has a bandwidth of 70 megabits per second. With the CI, any combination of processor nodes and intelligent I/O subsystem nodes — up to 16 in number — can be loosely coupled in a computer-room environment.

computer memory interconnect (CMI)

The part of the VAX–11/750 hardware that connects the processor, memory controllers, MASSBUS adapters, and the UNIBUS interconnect.

computing component

Part of the total computing system around which an arbitrary boundary has been defined. The boundary can be defined at any level.

concealed device

An I/O device that has a logical name associated with it; users thus see the logical name, rather than the device name, displayed in most system responses.

condition

An error state that exists when an exception occurs. See *exception* and *condition handler*.

condition codes

The 4 bits in the processor status word that indicate the results of previously executed instructions.

condition handler

A procedure that the system executes when a process exception occurs. When an exception occurs, the operating system searches for a condition handler and, if found, initiates the handler immediately. The condition handler may perform some action to change the situation that caused the exception and continue execution for the process that incurred the exception. Condition handlers execute in the context of the process at the access mode of the code that incurred the exception.

condition value

A 32-bit value that uniquely identifies the exception that caused the condition.

condition variable

A synchronization object used in conjunction with a mutex. A condition variable allows a thread to be blocked until some event happens.

configuration database

A database containing files that provide information about network components. Specifically, the files contain information about the local node, and all nodes, modules, circuits, lines, logging, and objects in the network. See also *permanent database* and *volatile database*.

configuration register

A control and status register (CSR) for an adapter, for example, a UNIBUS adapter. It resides in the adapter's I/O space.

congestion loss

A condition in which data packets transmitted over a network are lost when the DECnet–VAX Routing layer is unable to buffer them.

connect-to-interrupt

A function by which a process connects to a device interrupt vector. To perform a connect-to-interrupt, the process must map to the program I/O space containing the vector. See also *page frame number mapping*.

connection

The logical path in system communications architecture by which two processes communicate.

connector node

A node that serves as an X.25 gateway to permit VMS host nodes to access a packet switching data network.

consistent

A shadow set is consistent when every logical block on a member unit contains exactly the same data as the same logical block on the other members.

console

The manual control unit integrated into the central processor. The console enables the operator to start and stop the system, monitor system operation, and run diagnostics.

console terminal

The video or hardcopy terminal connected to the central processor console.

context

See *process context*.

context indexing

The ability to index through a data structure automatically because the size of the data type is known and used to determine the offset factor.

context switching

Interrupting the activity in progress and switching to another activity. Context switching occurs as one process after another is scheduled for execution. The operating system saves the hardware context of the interrupted process in its hardware process control block (PCB) using the Save Process Context instruction, and loads the hardware PCB of another process into the hardware context using the Load Process Context instruction, thus scheduling that process for execution.

contiguous area

A group of physically adjacent blocks on a device or in memory.

continuation character

A hyphen placed at the end of a command line, which allows the user to continue the command string on the next line after the Return key is pressed.

control key

The keyboard character that causes a control action. A control key is usually the combination of the Ctrl key and an alphabetic key, for example, Ctrl/Y.

control region

The highest-addressed half of per-process space (the P1 region). Control region virtual addresses refer to the process-related information used by the system to control the process, such as the kernel, executive, and supervisor stacks; the permanent I/O channels; exception vectors; and dynamically used system procedures (such as the command interpreter). The user stack is also normally found in the control region.

control region base register (P1BR)

The processor register, or its equivalent in a hardware process control block, that contains the base virtual address of a process control region page table.

control region length register (P1LR)

The processor register, or its equivalent in a hardware process control block, that contains the number of nonexistent page table entries for virtual pages in a process control region.

control region page table (P1PT)

The page table that maps the control region of virtual address space.

control station

The network node at the controlling end of a multipoint circuit. The control station controls the tributaries for that circuit.

control and status register (CSR)

A device or controller register residing in the processor's I/O space. The CSR initiates device activity and records its status.

controller

The part of a mass storage subsystem responsible for interfacing between drives and VAX computers. Controllers usually multiplex the services of one or more drives among the demands of one or more host computers.

controller data channel

A logical path to which a driver for a device on a multidevice controller must be granted access before it can activate a device.

cooperating tasks

Two tasks that communicate with each other in a task-to-task communication environment. In particular, cooperating tasks must agree on optional user data to be passed, how they will send and receive messages to ensure that there is one transmit for each receive, and which task will disconnect the link.

copy thread

In terms of volume shadowing, a copy thread is a single stream of I/O that is part of a full copy or merge operation.

copy-on-reference

A method used in memory management for sharing data until a process accesses it, in which case it is copied and made private before the access. Copy-on-reference allows sharing of the initial values of a global section whose pages have read/write access but contain preinitialized data available to many processes.

cost

An numeric value assigned to a circuit that exists between two adjacent nodes. In the DECnet network, data packets are routed on paths with the lowest cost.

counted string

A character-string data structure consisting of a byte-sized length followed by the string. Although a counted string is not used as a procedure argument, it is a convenient representation in memory.

counters

Performance and error statistics kept for a network component, such as lines or nodes.

CPU

See *central processing unit.*

crash

The system's response to an unstable condition, particularly if the system is corrupted. Rather than continuing to operate and possibly damaging itself, the system stops functioning. If the system has crashed, users will be unable to use their terminals. See also *hanging*.

crash dump

The action the operating system takes, when it crashes, to preserve information for future analysis. See also *complete crash dump* and *selective crash dump*.

CRB

See *channel request block*.

CRC

See *cyclic redundancy check*.

CRT

Cathode-ray tube. See also *terminal*.

CSMA/CD

See *Carrier Sense, Multiple Access with Collision Detect*.

CSR

See *control and status register*.

CUG

See *closed user group*.

current access mode

The processor access mode of the currently executing software indicated in the current mode field of the processor status longword.

cursor

An indicator used on a video terminal to point to the screen position where the next character will appear.

cyclic redundancy check (CRC)

An error detection scheme in which the receiver checks each block of data for errors. The check character is generated by taking the remainder after dividing all the serialized bits in a block of data by a predetermined binary number. The check character is compared with the transmitter-generated check character. If the check characters do not match, retransmission of the block of data is requested.

cylinder

The tracks at the same radius on all recording surfaces of a disk.

data

A general term referring to any representation of facts, concepts, or instructions in a form suitable for communication, interpretation, or processing.

data check

An operation that consists of first performing the derived physical, logical, or mutual read or write function successfully and then comparing the data in memory with the data on disk to make sure they match.

data circuit-terminating equipment (DCE)

A CCITT X.25 term referring to the network equipment that establishes, maintains, and terminates a connection and handles the signal conversion and coding between the data terminal equipment and the network. The switching exchange of the network to which DTEs (data terminal equipment) are connected. In non-X.25 usage, the term is synonymous with "modem."

data integrity

The ability of a system to maintain its information in a consistent state.

data link mapping (DLM)

Capability of using an X.25 virtual circuit as a DECnet data link.

data terminal equipment (DTE)

An X.25 term referring to the user's equipment (computer or terminal) connected to a DCE (data circuit-terminating equipment) on a packet switching data network (PSDN) for the purpose of sending and receiving data.

data structure

Any table, list, array, queue, or tree whose format and access conventions are well defined for reference by one or more images.

data type

In general, the way in which bits are grouped and interpreted. In reference to the processor instructions, the data type of an operand identifies the size of the operand and the significance of the bits in the operand. Operand data types include byte, word, longword, and quadword integer; floating and double-floating character string; packed decimal string; and variable-length bit field.

database

(1) All the occurrences of data described by a database management system.

(2) A collection of related data structures.

datagram

A unit of data sent over the network that is handled independently of all other units of data so far as the network is concerned. When a route header is added, a datagram becomes a packet.

DCE

See *data circuit-terminating equipment*.

DCL

See *DIGITAL Command Language*.

DDB

See *device data block*.

DDT

See *driver dispatch table*.

debug symbol table

The portion of the symbol table that is created by the compiler or assembler.

debugger

A program that aids a programmer in finding errors in other programs.

decryption

The process that restores encrypted information to its original unencoded form.

dedicated resource

A system resource—an I/O device, image, or the entire system—that is assigned to a single application or purpose.

default

A value or operation that is automatically included in a command, unless the user specifies otherwise. In most cases, default settings will be what is "normal" or "expected."

deferred echo

Refers to the fact that terminal echoing does not occur until a process is ready to accept input entered by type-ahead.

degraded performance

Performance in which service continues, but response time is extended or the number of users that can be served is reduced, or both.

delimiter

A character that separates, terminates, or organizes elements of a character string, statement, or program.

delta time

A time value expressing an offset from the current date and time. Delta times are always expressed in the system as negative numbers whose absolute value is used as an offset from the current time.

demand-zero page

A page, typically of an image stack or buffer area, that is initialized to contain all zeros when dynamically created in memory as a result of a page fault. This feature eliminates the waste of disk space that would otherwise be required to store blocks (pages) that contain only zeros.

density

The number of bits per inch (bpi or bpi/in) of magnetic tape. Typical values are 800 bpi and 1600 bpi. See *bits per inch*.

dependable computing system

One that can be counted on to provide services to its users when those services are needed, and with sufficient performance. The computing components are created and combined in the manner necessary to provide a required level of dependability.

descriptor

A data structure used in calling sequences for passing argument types, addresses, and other optional information. See *character string descriptor*.

designated router

A routing node on the Ethernet selected to perform routing services on behalf of end nodes.

destructor

A user-supplied routine that finalizes and then deallocates a per-thread context value.

detached process

A process that has no owner. The job controller creates a detached process when a user logs in to the system. It also creates a detached process each time it initiates a batch job or services a request for a logical link connect. Because the job controller does not own the processes it creates, these processes are referred to as detached. The DCL command RUN/UIC and the Create Process system service (specifying a UIC) allow a suitably privileged process to request creation of a detached process.

device

The general name for any peripheral connected to the processor that is capable of receiving, storing, or transmitting data. Card readers, line printers, and terminals are examples of record-oriented devices. Magnetic tape devices and disk devices are examples of mass storage devices. Terminal line interfaces and interprocessor links are examples of communications devices. Devices are not necessarily hardware (see also *pseudodevice*).

device affinity

In a VMS symmetric multiprocessing system, a close association of a device with a specific processor or set of processors in the system. There are three dimensions to device affinity in a VMS system. First, physical connectivity describes those devices that are directly accessible only to the primary processor or to all processors. Secondly, affinity is a software mechanism that defines those processors that can initiate an I/O operation on the device. Finally, interruptibility describes the set of processors that can receive interrupts from a device.

device allocation

See *allocate*.

device controller

The electronic circuits associated with each physical device in the system that serve as the interface between the processor and the device hardware.

device data block (DDB)

A structure in the I/O database that identifies the generic device or controller name and driver name for a set of devices attached to the same controller.

device driver

The software associated with each physical device in the system that serves as the interface between the operating system and the device controller.

Device driver code is divided into two sections: device-dependent logic that drives a particular kind of device unit, and device-independent support routines that perform functions common to all devices. Each unit of a particular type has a driver process, but there is only one set of driver code.

device interrupt

An interrupt received on interrupt priority levels (IPLs) 20 through 23. Device interrupts can be requested only by devices, controllers, and memories.

device lock

In a VMS symmetric multiprocessing system, a dynamic spin lock, the ownership of which synchronizes device-specific code that executes at device interrupt priority level (IPL). A device lock is associated with each adapter or controller in the system. See *spin lock*.

device name

The field in a file specification that identifies the device unit on which a file is stored. Device names also include the mnemonics that identify an I/O peripheral device in a data transfer request. A device name consists of a mnemonic followed by a controller identification letter (if applicable), followed by a unit number (if applicable), and ends with a colon.

device queue

See *spool queue*.

device register

A location in device controller logic used to request device functions (such as I/O transfers) or report status.

device unit

An I/O device and its controlling logic; for example, a disk drive or terminal. Some controllers can have several device units connected to a single controller; for example, mass storage controllers.

D_floating point data

See *double floating data*.

diagnostic

A program that tests hardware, firmware, peripheral operation, logic, or memory and reports any faults it detects.

DIGITAL Command Language (DCL)

A command interpreter in a VMS system. It provides a means of communication between the user and the operating system. Contrast with *monitor console routine*.

DIGITAL Network Architecture (DNA)

A set of protocols (rules) governing the format, control, and sequencing of message-exchange for all Digital network implementations. The protocols are layered, and they define rules for data exchange from the physical link level up through the user interface level. DNA controls all data that travels throughout a Digital network. DNA also defines standard network management and network generation procedures.

Digital Storage Architecture (DSA)

The specifications from Digital governing the design of and interface to mass storage products. DSA defines the functions to be performed by host computers, controllers, and drives, and specifies how they interact to manage mass storage.

Digital Storage Systems Interconnect (DSSI)

A data bus that uses the System Communication Architecture (SCA) protocols for direct host-to-storage communications. The DSSI cable can extend to 6 meters and has a peak bandwidth of 4 megabytes.

direct data path

A UNIBUS adapter data path that transfers 16 bits of data in a single synchronous backplane interconnect (SBI) transfer.

direct I/O

An I/O operation in which the system locks the pages containing the associated buffer in physical memory for the duration of the I/O operation. The I/O transfer takes place directly from the process buffer. Contrast with *buffered I/O*.

direct mapping cache

A cache organization in which only one address comparison is needed to locate any data in the cache because any block of main memory data can be placed in only one possible position in the cache. Contrast with *fully associative cache*.

direct memory access

The method by which a device driver transfers a large amount of data without requesting an interrupt after transferring each of the smaller amounts.

directory

A file that briefly catalogs a set of files stored on disk or tape. The directory includes the name, type, and version number of each file in the set, as well as a unique number that identifies the file's actual location and points to a list of its attributes. See also *master file directory* and *subdirectory*.

directory name

The field in a file specification that identifies the directory in which a file is listed. The directory name begins with a left bracket ([or <) and ends with a right bracket (] or >).

disconnect abort

A form of disconnection by which nontransparent tasks can deaccess a logical link without deassigning the channel. A disconnect abort indicates that not all messages sent have necessarily been received.

discretionary controls

Security controls that are applied at the user's option; that is, they are not required. Access control lists are typical of such optional security features.

disk scavenging

Any method of obtaining information from a disk that the owner intended to discard. The information, although no longer accessible by normal means, retains a sufficient amount of its original magnetic encoding so that it can be retrieved and used by one of the scavenging methods.

displacement deferred indexed mode

An indexed addressing mode in which the base operand specified uses displacement deferred mode addressing. See also *indexed addressing mode* and *displacement deferred mode*.

displacement deferred mode

An addressing mode in which the specifier extension is a byte, word, or longword displacement. The displacement is sign-extended to 32 bits and added to a base address obtained from the specified register. The result is the address of a longword that contains the address of the actual operand. If the PC is used as the register, the updated contents of the PC are used as the base address. The updated contents of the PC then become the address of the first byte beyond the specifier extension.

displacement indexed mode

An indexed addressing mode in which the base operand specifier uses displacement mode addressing. See also *indexed addressing mode* and *displacement mode*.

displacement mode

An addressing mode in which the specifier extension is a byte, word, or longword displacement. The displacement is sign extended to 32 bits and added to a base address obtained from the specified register. The result is the address of the actual operand. If the PC is used as the register, the updated contents of the PC are used as the base address. The updated contents of the PC then become the address of the first byte beyond the specifier extension.

distributed transaction processing

The processing of transactions on remote nodes. A user logged in to a transaction processing system on one node can select tasks in an application on a transaction processing system on another node.

DLM

See *data link mapping*.

DMA

See *direct memory access*.

DNA

See *DIGITAL Network Architecture*.

double floating data

A double precision floating-point number, eight bytes long, having a range of +/– $2.9 * 10^{-37}$ to +/– $1.7 * 10^{38}$ and a precision of approximately 16 decimal digits. See also *D_floating data* and *G_floating data*.

downline system load

A DECnet VAX function that allows an unattended target node to receive an operating system file image or terminal server image from another node.

downline task load

A DECnet VAX function that allows a remote target node to receive an RSX–11S task from another node.

down time

The percentage or amount of time a computing system does not provide application service as scheduled.

DPT

See *driver prolog table*.

drive

The electromechanical unit of a mass storage device system on which a recording media (disk cartridge, disk pack, or magnetic tape reel) is mounted.

driver

See *device driver*.

driver code

See *device driver*.

driver dispatch table (DDT)

A table in a driver that lists the entry point addresses of standard driver routines and the sizes or diagnostic and error logging buffers for the device type.

driver fork level

The interrupt priority levels (IPLs) at which a driver fork processes executes, that is, IPLs 8 through 11. Every unit control block indicates the address of a fork lock that synchronizes driver resources. The fork lock structure indicates the fork level or driver.

driver prolog table (DPT)

A table in the driver that describes the driver and the device type to the VMS procedure that loads drivers into the system.

driver start I/O routine

See *start I/O routine*.

DSA

See *Digital Storage Architecture*.

DSSI

See *Digital Storage Systems Interconnect*.

DST

See *debug symbol table*.

DTE

See *data terminal equipment*.

DV

The bit in the processor status word that indicates that decimal overflow traps are enabled.

dynamic access

A technique in which a program switches from one record access mode to another while processing a file.

dynamic load balancing

A method of work distribution in which the operating system ensures that the system work load is evenly distributed among the processors. Dynamic load balancing in a VMS symmetric multiprocessing system is a direct effect of the implementation of the scheduler. In a VMS multiprocessing system, processors independently and continually look for processes to execute from a common pool of processes.

EBCDIC

See *Extended Binary Coded Decimal Interchange Code*.

ECB

See *exit control block*.

ECC

See *error correction code*.

echo

A terminal handling characteristic in which the characters typed by the user on the terminal keyboard are also displayed on the screen or printer.

editor

A system image used for creating and altering text files.

effective address

The address obtained after deferred or indexing modifications are calculated.

encryption

A process of encoding information so that its content is no longer immediately obvious to anyone who obtains a copy of it.

end node

A node that can receive packets addressed to it and send packets to other nodes, but cannot route packets through from other nodes. Also called a nonrouting node.

entry mask

A word whose bits represent the registers to be saved when a procedure is called with a CALLS or CALLG instruction and restored when the procedure executes a RET instruction.

entry point

A location that can be specified as the object of a call. It is identified by a symbolic name and contains a mask known as the register save mask that determines which registers are saved before the procedure is called and can enable DV or IV arithmetic traps.

equivalence name

The string associated with a logical name in a logical name table. An equivalence name can be, for example, a device name, another logical name, or a logical name concatenated with a portion of a file specification. See also *search list*.

erase-on-allocate

A technique that applies an erasure pattern whenever a new area is allocated for a file's extent. The new area is erased with the erasure pattern so that subsequent attempts to read the area can only yield the erasure pattern and not some valuable remaining data. This technique is used to discourage disk scavenging.

erase-on-delete

A technique that applies an erasure pattern whenever a file is deleted or purged. This technique is used to discourage disk scavenging.

erasure pattern

A character string that can be used to overwrite magnetic media for the purpose of erasing the information that was previously stored in that area.

error

An event during the operation of a computing component that produces incorrect results due to one or more faults. Errors are observed as incorrect responses within a specific computing component.

error correction

The action necessary to isolate the effects of faults to a specific computing component. The goal is to contain the impact of the problem.

error correction code (ECC)

Code that carries out automatic error correction by performing an exclusive OR operation on the transferred data and applying a correction mask.

error logger

A system process that empties the error log buffers and writes the error messages into the error file. Errors logged by the system include memory system errors, device errors and timeouts, and interrupts with invalid vector addresses.

escape sequence

An escape is a transition from the normal mode of operation to a mode outside the normal mode. An ASCII escape character is the code that indicates the transition from normal to escape mode. An escape sequence refers to the set of character combinations starting with an escape character that the terminal transmits without interpretation to the software set up to handle escape sequences.

ESP

See *executive-mode stack pointer.*

ESR

See *exception service routine.*

Ethernet

A communications concept for local communication networks that employs coaxial cable as a passive communications media to interconnect different kinds of computers, information processing products, and office equipment at a local business site. Ethernet does not require switching logic or control by a central computer.

evasive action

A responsive behavior by VMS to discourage break-in attempts whenever they seem to be in progress. VMS has a set of criteria it uses to detect the fact that break-in attempts may be underway. Typically, once VMS suspects that an unauthorized user is attempting to log in, the evasive action consists of locking out all login attempts by the offender for a limited period of time.

event

(1) A change in process status or an indication of the occurrence of some activity that concerns an individual process or cooperating processes; an incident reported to the scheduler that affects a process's ability to execute. Events can be synchronous with the process's execution (a wait request), or they can be asynchronous (I/O completion). Some other events include swapping, wake request, and page fault.

(2) A network or system-specific occurrence for which the logging component maintains a record.

(3) An exception or interrupt.

event class

A particular classification of events. Generally, this classification follows the DIGITAL Network Architecture (DNA) architectural layers; some layers may contain more than one class. Class also includes the identification of system-specific events.

event flag

A bit in an event-flag cluster that can be set or cleared to indicate the occurrence of the event associated with that flag. Event flags are used to synchronize activities in a process or among many processes.

event-flag cluster

A set of 32 event flags used for event posting. Four clusters are defined for each process: two process-local clusters and two common event flag clusters. Of the process-local flags, eight are reserved for system use.

event type

A particular form of event that is unique within an event class.

exception

An event detected by the hardware or software (other than an interrupt or jump, branch, case, or call instruction) that changes the normal flow of instruction execution. An exception is always caused by the execution of an instruction or set of instructions (whereas an interrupt is caused by an activity in the system independent of the current instruction). There are three types of hardware exceptions: traps, faults, and aborts. Examples are attempts to execute a privileged or reserved instruction, trace traps, compatibility mode faults, breakpoint instruction execution, and arithmetic traps such as overflow, underflow, and division by zero.

exception dispatcher

An operating system procedure that searches for a condition handler when an exception condition occurs. If no exception handler is found for the exception or condition, the image that incurred the exception is terminated.

exception enables

See *trap enables*.

exception vector

See *vector*.

exception service routine

The routine by which VAX hardware initially passes control to service an exception. An exception service routine passes control to a general exception dispatcher that attempts to locate a condition handler to further service the exception.

executable image

An image that can be run in a process. When run, an executable image is read from a file for execution in a process.

executive

The generic name for the collection of procedures included in the operating system software that provides the operating system's basic control and monitoring functions.

executive mode

The second most privileged processor access mode (mode 1). The VMS Record Management Services (RMS) and many of the operating system's system service procedures execute in executive mode.

executive-mode stack pointer (ESP)

The process context stack pointer for executive mode.

executor node

The node at which a Network Control Program (NCP) command actually executes.

exit

An image rundown activity that occurs when image execution terminates either normally or abnormally. Image rundown activities include deassigning I/O channels and disassociating common event flag clusters. Any user- or system-specified exit handlers are called.

exit control block

A data structure that describes an exit handling routine declared by the Declare Exit Handler system service, $DCLEXH.

exit handler

A procedure executed when an image exits. An exit handler enables a procedure that is not on the call stack to gain control and clean up procedure-owned databases before the actual image exit occurs.

extended attribute block (XAB)

A VMS Record Management Services (RMS) user data structure that contains additional file attributes beyond those expressed in the file access block (FAB), such as boundary types (aligned on cylinder, logical block number, and virtual block number) and file protection information.

Extended Binary Coded Decimal Interchange Code (EBCDIC)

A set of 8-bit characters used for representing data. See also *ASCII*.

extended QIO processor (XQP)

A facility that supplements the QIO driver's functions when performing virtual I/O operations on file-structured devices (Files–11 On-Disk Structure Level 2). The XQP executes in kernel mode in the context of the process of its caller.

extension

The amount of space allocated at the end of a file each time a sequential write exceeds the allocated length of the file.

extent

The contiguous area on a disk containing a file or a portion of a file. An extent consists of one or more clusters.

F11ACP

See *Files–11 ancillary control process*.

FAB

See *file access block*.

failover

(1) The process of reconfiguration after a hard fault or for planned maintenance.

(2) The ability of a system or component to reconfigure itself.

failure

The inability of a computing component to perform its function correctly due to one or more internal faults whose effects cannot be contained. Failures are observed as incorrect behavior by the consumers of the computing component's services.

failure exception mode

A mode of execution selected by a process that instructs the system to declare an exception condition if an error occurs as the result of a system service call. The normal mode is for the system service to return a status code for which the process must test.

failure recovery

The action necessary to restore the failed computing component to a correctly functioning condition. The goal is prompt return to zero defects.

fallback

A function that the Terminal Fallback Facility (TFF) performs when an application program sends a character that a terminal cannot display. In this case, TFF replaces the character with the closest possible visual character that the terminal can display.

fault

(1) A hardware exception condition that occurs in the middle of an instruction and leaves the registers and memory in a consistent state so that eliminating the fault and restarting the instruction gives correct results.

(2) A defect in some component of a computing system.

fault management

The discipline used to engineer systems with a cost-effective balance of fault prevention qualities, error correction capabilities, and failure recovery facilities. Fault management is realized in the implementation of a dependable computing system. It is also a philosophy that is followed during the implementation.

fault prevention

The process of designing and constructing computing components to be free from faults. The goal is zero defects.

fault tolerance

(1) The ability of a computing system to sustain a single operational fault while it continues to provide service without user intervention, with no significant delay in service, without loss of work in progress, and with complete data integrity.

(2) The ability of a computing system to withstand faults and errors while continuing to provide the required services. See also *hardware-based fault tolerance* and *software-based fault tolerance*.

FCB

See *file control block*.

FCS

See *file control system*.

FDL

See *file definition language*.

FDT

See *function decision table*.

FDT routine

A driver routine called by the Queue I/O Request system service to perform device-dependent preprocessing of an I/O request.

F_floating-point data

See *floating (point) data*.

FID

See *file identifier*.

field

A set of contiguous bytes in a logical record. See also *variable-length bit field*.

FIFO

First-in/first-out; the order in which processing is performed. For example, processing on a FIFO queue would be on a first-come, first-served basis. See also *LIFO*.

file

A set of data elements arranged in a structure significant to the user. A file is any named, stored program or data, or both, to which the system has access. Access can be of two types: read-only, meaning the file is not to be altered, and read-write, meaning the contents of the file can be altered. See also *volume*.

file access block (FAB)

A VMS Record Management Services (RMS) user data structure that describes a particular file and contains file-related information needed for data operations, such as OPEN, CLOSE, or CREATE.

file control block (FCB)

A memory-resident structure used by the operating system to coordinate access to a file that is opened by an active task.

file control system (FCS)

A generic term for the file control services that control the file services for disks and tapes. These services include adding, opening, closing, and renaming files.

file definition language (FDL)

A special-purpose language used to write specifications for data files. These specifications are written in text files called FDL files; they are then used by VMS Record Management Services (RMS) utilities and library routines to create the actual data files.

file header

A block in the index file describing a file on a Files–11 disk structure. The file header contains information needed by the file system to find and use the file. Some of this information is displayed when the DCL command DIRECTORY is entered. There is at least one file header for every file on the disk.

file identifier (FID)

A 6-byte value used to uniquely identify a file on a Files–11 disk volume. The file number, file sequence number, and relative volume number are contained in the file identifier.

file name

The field containing a 1- to 39-character name for a file that precedes a file type in a file specification.

file name extension

See *file type*.

file organization

The particular file structure used as the physical arrangement of the data comprising a file on a mass storage media. The VMS Record Management Services (RMS) file organizations are sequential, relative, and indexed.

file section

The part of a file that contains the user data and that is delimited by the header and trailer labels. Only one section of a given file can be written on any one volume. Multiple sections of a file or other file sections cannot be interspersed within a file section.

file sharing

The capability of a particular relative or indexed file to allow access to more than one process.

file specification

A unique name for a file on mass storage media. It identifies the node, the device, the directory name, the file name, the file type, and the version number under which a file is stored.

file structure

The way in which the blocks forming a file are distributed on a disk or magnetic tape.

file system

A method of recording, cataloging, and accessing files on a volume.

file type

The field in a file specification that consists of a period followed by a 0- to 39-character identification. By convention, this field identifies a generic class of files that have the same use or characteristics, such as compiler and assembler listing files, binary object files, and so on.

Files–11

The name of the disk structure used by the RSX–11, IAS, and VMS operating systems. See *Files–11 On-Disk Structure Level 1* and *Files–11 On-Disk Structure Level 2*.

Files–11 ancillary control process (F11ACP)

The interface process that is the files manager for the Files–11 on-disk structure.

Files–11 On-Disk Structure Level 1

The original Files–11 structure used by IAS, RSX–11M, and RSX–11D for disk volumes. VMS supports structure level 1 to ensure compatibility among systems.

Files–11 On-Disk Structure Level 2

The second-generation disk file structure supported by VMS. The Files–11 data structure prepares a volume to receive and store data in a way recognized by the VMS operating system.

first part of an instruction done (FPD)

A bit in the processor status longword. If the FPD is set, when the processor returns from an exception or interrupt, it resumes the interrupted operation where it left off, rather than restarting the instruction.

fixed-length control area

An area, prefixed to a variable-length record, containing additional information about the record that may have no bearing on the other contents of the record. The fixed-length control area may be used, for example, to contain line numbering or carriage control information.

fixed-length record format

A file format in which all records have the same length.

fixed line number

A number fixed to a line of text in a file. The EDT text editor maintains a record of line numbers during the editing sessions in which the files are used; they are copied to the file when the editing session ends.

flag

A bit that can be set to invoke the execution of a particular sequence of instructions; frequently, an indicator used to tell some later part of a program that a certain condition occurred earlier.

floating (point) data

A single precision floating-point number, 4 bytes long, having a range of +/– $2.9 * 10^{-37}$ to +/– $1.7 * 10^{38}$ and a precision of approximately seven decimal digits.

floating underflow (FU)

A trap enable bit in the processor status word (PSW).

foreign volume

Any volume other than a Files–11 formatted volume. A foreign volume may or may not be file-structured.

fork block

The portion of a unit control block that contains a driver's context while the driver is waiting for a resource, or the portion of another structure used as a fork block to enable the driver to synchronize events at a lower interrupt priority level (IPL). A driver awaiting a processor or adapter resource has its fork block linked into a fork queue.

fork dispatcher

A VMS interrupt service routine that is activated by a software interrupt at a fork interrupt priority level (IPL). Once activated, it dispatches driver fork processes from a processor-specific fork queue until no processes remain in the queue for that IPL.

fork lock

In a VMS symmetric multiprocessing system, a static spin lock, the ownership of which synchronizes the right of a driver's fork process to execute at its associated fork interrupt priority level (IPL). See also *spin lock*.

fork process

A minimal context process that executes code under a series of constraints: it executes at raised interrupt priority levels; it uses R0 through R5 only (other registers must be saved and restored); it executes in system virtual address space; it is only allowed to refer to and modify static storage that is never modified by higher interrupt priority level code. The VMS operating system uses software interrupts and fork processes to synchronize executive operations.

fork queue

A processor-specific queue of driver fork blocks that is awaiting activation, at a particular interrupt priority level, by the VMS fork dispatcher.

form feed

The movement of the cursor position to the start of a new page.

FP

See *frame pointer*.

FPD

See *first part of an instruction done*.

frame

A unit, which is delimited by flags and includes a header, used by the link level to exchange data packets as well as control and error information between data terminal equipment (DTE) and data circuit-terminating equipment (DCE) on a packet switching data network. See also *stack frame*.

frame pointer (FP)

General register 13 (R13). By convention, it contains the base address of the most recent call frame on the stack.

front-end

The part of a computing system that typically handles data capture, terminal displays, communications, and validation functions.

FU

See *floating underflow*.

full copy member

A device that is not yet a member of the shadow set, but that is being added to the shadow set and requires a full copy operation because it is completely inconsistent with respect to the other members in the shadow set.

full copy operation

Copies data from an active shadow set member to a new, specified volume in order to make the new device consistent and bring it into full shadow set membership. The specified volume should be a blank volume or a former shadow set member that is not consistent with the other shadow set members.

fully associative cache

A cache organization in which any block of data from main memory can be placed anywhere in the cache. Address comparison must take place against each block in the cache to find any particular block. Contrast with *direct mapping cache*.

function code

See *I/O function code*.

function decision table (FDT)

A table in a device driver that lists all valid function codes for the device and the addresses of I/O preprocessing routines associated with each valid function.

function modifier

See *I/O function modifier*.

general identifier

One of three possible types of identifiers that specify one or more groups of users. The general identifier is alphanumeric and typically is a convenient term that symbolizes the nature of the group of users. For example, a typical general identifier might be PAYROLL for all users allowed to run payroll applications.

general register

Any of the sixteen 32-bit registers used as the primary operands of the native-mode instructions. The general registers include 12 general-purpose registers, which can be used as accumulators, as counters, and as pointers to locations in main memory, and the FP, AP, SP, and PC.

generic device name

A device name that identifies the type of device but not a particular unit; a device name in which the specific controller or unit number is omitted.

G_floating data

An extended-range floating-point number, 8 bytes long, having a range of +/– $0.56 * 10^{-308}$ to +/– $0.9 * 10^{308}$ and a precision of approximately 15 decimal digits.

global

Affecting the entire file, the entire system, or the entire image, depending on context. A global substitution in a text file would be the changing of all instances of a particular string to something else, for example, changing "dog" to "cat."

global page table

The page table containing the master page table entries for global sections.

global section

A data structure (for example, FORTRAN global common) or shareable image section potentially available to all processes in the system. Access is protected by privilege or group number of the user identification code (UIC), access control list (ACL), or both.

global symbol

(1) A symbol defined in a module of a program that is potentially available for reference by another module. The linker resolves (matches references with definitions) global symbols. Contrast with *local symbol*.

(2) A command language symbol accessible at all command levels.

global symbol table (GST)

In a library, an index of strongly defined global symbols used to access the modules defining the global symbols. The linker also puts global symbol tables into an image. For example, the linker appends a global symbol table to executable images that are intended to run under the symbolic debugger, and it appends a global symbol table to all shareable images.

Gold key

The upper-left key on the VT100 series terminal keypad, which enables alternate keypad functions.

GROUP

(1) In the context SYSTEM/OWNER/GROUP/WORLD, a set of users who have special access privileges to each other's directories and files within those directories (unless protected otherwise). GROUP refers to all accounts having the same group number as a particular owner.

(2) A set of jobs (processes and their subprocesses) with access to a group's common event flags and logical name tables.

group number

The first number or its alphanumeric equivalent in a user identification code (UIC).

GST

See *global symbol table*.

handshaking sequence

The exchange of logical link connection information between two tasks. This exchange takes place to enable the successful completion of a logical link connection.

hanging

The extremely slow response of the system. During an interactive session, when the terminal fails to respond, it is said to be hanging. When the system hangs, users might think that it has crashed.

hardcopy terminal

Terminals that print output on paper. See also *terminal*.

hardware address

For an Ethernet device, the unique Ethernet physical address associated with a particular Ethernet communications controller (usually in read-only memory) by the manufacturer.

hardware-based fault tolerance

The ability to detect, isolate, and bypass a fault that has been engineered into and executed through hardware.

hardware context

The values contained in the following registers while a process is executing:

- The program counter (PC)
- The processor status longword (PSL)
- The 14 general registers (R0 through R13)
- The four processor registers (P0BR, P0LR, P1BR and P1LR) that describe the process's virtual address space
- The stack pointer (SP) for the access mode in which the processor is executing
- The contents to be loaded in the SP for every access mode other than the current access mode

When a process is executing, its hardware context is continually being updated by the processor. When a process is not executing, its hardware context is stored in its hardware process control block (PCB).

hardware process control block (hardware PCB)

A data structure known to the processor that contains the saved hardware context when a process is not executing. A process's hardware PCB resides in its process header.

hibernation

A state in which a process is inactive, but known to the system. A hibernating process becomes active again when a wake request is issued. It can schedule a wake request before hibernating, or another process can issue its wake request. A hibernating process can also become active long enough to service any AST it may receive while it is hibernating. Contrast with *suspension*.

hierarchical storage controller (HSC)

A self-contained, intelligent, mass-storage controller that communicates with VAX processors and implements volume shadowing on Digital Storage Architecture (DSA) disks.

high-level language

A language for specifying computing procedures or organization of data within a digital computer. High-level languages are distinguished from assembly and machine languages by the omission of machine-specific details required for direct execution on a given computer.

highwater marking

A technique for discouraging disk scavenging. The system tracks the furthest extent that the owner of a file has written into the file's allocated area. It then prohibits any attempts at reading beyond the written area on the premise that any information that exists beyond the currently written limit is information some user had intended to discard. The VMS operating system accomplishes the goals of highwater marking with its erase-on-allocate strategy.

H_floating data

An extended range floating-point number, 16 bytes long, having a range of +/– $0.84 * 10^{-4932}$ to +/– $0.59 * 10^{4932}$ and a precision of approximately 33 decimal digits.

holder

A user who possesses a particular identifier. The user is said to be the holder of an identifier if he or she possesses that identifier. The rights database is the place where the system associates users and the identifiers they hold.

home block

A block in the index file that contains the volume identification, such as volume label and protection.

hop

The logical distance between two nodes. One hop is the distance from one node to an adjacent node.

host node

(1) For a DECnet network, a node that provides services for another node (for example, the host node supplies program image files for a downline load). For the VAX Packetnet System Interface (PSI), a node that accesses a packet switching data network by means of an X.25 multihost connector node.

(2) The node that makes a device available to other nodes in a VAXcluster configuration. A host node can be either a VAX processor that adds the device to the mass storage control protocol (MSCP) server database or a hierarchical storage controller (HSC) server.

hot standby

A second running computing system that is ready to pick up application processing in the event the primary computing system fails.

HSC

See *hierarchical storage controller*.

IDB

See *interrupt dispatch block*.

identifier

A notation that represents a user or group of users to the system. There are three types of identifiers: user identification code (UIC) identifiers, system-defined identifiers, and general identifiers. Identifiers are stored in the rights database.

image

Procedures and data bound together by the linker to form an executable program. This executable program is executed by the process. There are three types of images: executable, shareable, and system.

image activator

A set of system procedures that prepares an image for execution. The image activator establishes the memory management data structures required both to map the image's virtual pages to physical pages and to perform paging.

image exit

See *exit*.

image I/O segment

That portion of the control region that contains the VMS Record Management Services (RMS) internal file access blocks (IFAB) and I/O buffers for the image currently being executed by a process.

image name

The name of the file in which an image is stored.

image privileges

The privileges assigned to an image when it is installed. See also *process privileges*.

image section (ISECT)

A group of program sections (PSECTs) with the same attributes (such as read-only access, read/write access, absolute, relocatable, and so on) that is the unit of virtual memory allocation for an image.

immediate mode

An addressing mode in which the program counter (PC) is used as the register in autoincrement mode addressing.

inbound connection

Logical link connection requests that a task receives.

increment

(1) To add one quantity to another.

(2) The quantity added.

index

The structure that allows retrieval of records in an indexed file by key value. See *key (indexed files)*.

index file

The file on a Files–11 volume that contains the access information for all files on the volume and enables the operating system to identify and access the volume.

index file bitmap

A table in the index file of a Files–11 volume that indicates which file headers are in use.

index register

A register used to contain an address offset.

indexed addressing mode

An addressing mode in which two registers are used to determine the actual instruction operand: an index register and a base operand specifier. The contents of the index register are used as an index (offset) into a table or array. The base operand specifier supplies the base address of the array (the base operand address or BOA). The address of the actual operand is calculated by multiplying the contents of the index register by the size (in bytes) of the actual operand and adding the result to the base operand address. The addressing modes resulting from index mode addressing are formed by adding the suffix "indexed" to the addressing mode of the base operand specifier: register deferred indexed, autoincrement indexed, autoincrement deferred indexed (or absolute indexed), autodecrement indexed, displacement indexed, and displacement deferred indexed.

indexed file organization

A file organization in which a file contains records and a primary key index (and optionally one or more alternate key indexes) used to process the records sequentially by index or randomly by index.

indirect command file

See *command procedure*.

input stream

The source of commands and data—the user's terminal, the batch stream, or a command procedure.

instruction buffer

An 8-byte buffer in the processor used to contain bytes of the instruction currently being decoded and to prefetch instructions in the instruction stream. The control logic continuously fetches data from memory to keep the 8-byte buffer full.

integrated storage element (ISE)

A generation of storage devices in which the controller and the device are in the same box.

interactive system

A computer system in which the user and the operating system communicate directly by means of a terminal. The operating system immediately acknowledges and acts upon requests entered by the user at the terminal.

interleaving

Assigning consecutive physical memory addresses alternately between two memory controllers.

interprocess communication facility

A common event flag cluster, mailbox, or global section used to pass information between two or more processes.

interrecord gap (IRG)

A blank space deliberately placed between data records on the recording surface of a magnetic tape.

interrupt

(1) An event other than an exception or a branch, jump, case, or call instruction that changes the normal flow of instruction execution. Interrupts are generally external to the process executing when the interrupt occurs. See also *device interrupt*, *software interrupt*, and *urgent interrupt*.

(2) A packet, sent through a packet switching data network, that bypasses normal flow control procedures used by data packets.

interrupt dispatch block (IDB)

A structure in the I/O database that describes the characteristics of a particular controller and points to devices attached to that controller.

interrupt message

A user-generated message sent outside the normal exchange of data messages during nontransparent DECnet task-to-task communication. This use of the term interrupt is contrary to the normal usage, which means to designate a software or hardware interrupt mechanism.

interrupt priority level (IPL)

The interrupt level at which a software or hardware interrupt is generated. There are 31 possible interrupt priority levels: IPL 1 is lowest, 31 is highest. The levels arbitrate contention for processor service. For example, a device cannot interrupt a processor if the processor is currently executing at an interrupt priority level equal to or greater than the interrupt priority level of the device's interrupt service routine.

interrupt service routine (ISR)

The routine executed when an interrupt occurs.

interrupt stack (IS)

The processor-specific stack used when executing in interrupt service context. At any time, the processor is either in a process context executing in user, supervisor, executive, or kernel mode, or in interrupt service context operating in kernel mode, as indicated by the interrupt stack and current mode bits in the processor status longword. The context of the interrupt stack is not switched.

interrupt stack pointer (ISP)

The stack pointer for the systemwide interrupt stack.

interrupt vector

See *vector*.

I/O database

A collection of data structures that describes I/O requests, controllers, device units, volumes, and device drivers in a VMS system. Examples are the driver dispatch table, driver prolog table, device data table, unit control block, channel request block, I/O request packet, and interrupt dispatch block.

I/O driver

See *device driver*.

I/O function

An I/O operation interpreted by the operating system and typically resulting in one or more physical I/O operations.

I/O function code

A 6-bit value specified in a Queue I/O Request system service that describes the particular I/O operation to be performed (for example, read, write, rewind).

I/O function modifier

A 10-bit value specified in a Queue I/O Request system service that modifies an I/O function code (for example, read terminal input no echo).

I/O lockdown

The state of a page when it cannot be paged or swapped out of memory.

I/O request packet (IRP)

A structure in the I/O database that describes an individual I/O request. The Queue I/O Request system service creates an I/O request packet for each I/O request. The VMS operating system and the driver of the target device use information in the I/O request packet to process the request.

I/O rundown

An operating system function in which the system cleans up any I/O in progress when an image exits.

I/O space

The region of physical address space that contains the configuration registers, and device control and status and data registers. These regions are not physically contiguous.

I/O status block (IOSB)

A data structure associated with the Queue I/O Request system service. This service optionally returns a status code, number of bytes transferred, and device- and function-dependent information in an IOSB. An IOSB is not returned from the service call, but filled in when the I/O request completes.

IPL

See *interrupt priority level*.

IRG

See *interrecord gap*.

IRP

See *I/O request packet*.

IS

See *interrupt stack*.

ISECT

See *image section*.

ISP

See *interrupt stack pointer*.

ISR

See *interrupt service routine*.

IV

Integer overflow trap enable bit in the processor status word.

job

The accounting unit equivalent to a process and its subprocesses, if any, and all subprocesses that they create. Jobs are classified as batch and interactive. For example, the job controller creates an interactive job to handle a user's requests when the user logs in to the system, and it creates a batch job when the symbiont manager passes a command input file to it.

job controller

The system process that establishes a job's process context, starts a process running the LOGIN image for the job, maintains the accounting record for the job, manages symbionts, and terminates a process and its subprocesses.

job information block (JIB)

A data structure associated with a job that contains the quotas pooled by all processes in the job.

journal file

A file containing the data input to the terminal for one editing session.

journaling

The recording of input during an editing session.

K

A unit for measuring the size of memory or similar resources. K is short for kilo and is used to mean roughly 1000, although K is equal to 2^{10}, or 1024.

kernel mode

The most privileged processor access mode (mode 0). The operating system's most privileged services, such as I/O drivers and the pager, run in kernel mode.

kernel-mode stack pointer (KSP)

The process context stack pointer for kernel mode.

key

(1) In indexed files, a character string, a packed decimal number, a 2- or 4-byte unsigned binary number, or a 2- or 4-byte signed integer within each data record in an indexed file. The user defines the length and location within the records; VMS Record Management Services (RMS) uses the key to build an index. See *primary key*, *alternate key*, and *random access by key value*.

(2) In relative files, the relative record number of each data record in a data file; VMS Record Management Services (RMS) uses the relative record numbers to identify and access data records in a relative file in random access mode. See *relative record number*.

(3) In the Sort Utility, the data field in a record that contains the information by which the user wants to sort the records.

keypad

The small set of keys next to the main keyboard on a terminal.

keyword

A word reserved for use in certain specified syntax formats, usually in a command string or a statement.

known component

The classification for one or more of the same DECnet components. This classification includes all active and inactive occurrences of the component type. For example, known nodes include all active and inactive nodes in the network.

KSP

See *kernel-mode stack pointer*.

label

A record that identifies and delimits a magnetic tape volume or file section.

label group

A collection of one or more contiguous label sets.

label identifier

The first three characters of a label name, which identify one or more labels within a label set. These characters will always be the same; for example, a file identifier of HDR will always be used to identify header labels within a header label set.

label number

A number that indicates the position of a label within a label set. For ANSI labels, label number 1 is always present if the label set exists.

label set

One or more contiguous labels on a magnetic tape volume or file section with the same label identifier.

LAT

See *local area transport*.

LAP

See *link access protocol*.

LBN

See *logical block number*.

LCN

See *logical channel number*.

level 1 router

A DECnet node that can send and receive packets, and route packets from one node to another node within a single area.

level 2 router

A DECnet node that can send and receive packets, and route packets from one node to another within its own area and between areas. Also known as an *area router*.

lexical function

A command language construct that the DIGITAL Command Language (DCL) command interpreter evaluates and substitutes before it parses a command string. Lexical functions return information about the current process (the user identification code [UIC] or default directory, for example) and about character strings, (their length or the location of substrings, for example).

librarian

A program that allows the user to create, update, modify, list, and maintain object library, help library, text library, and assembler macro library files.

library file

A direct access file containing one or more modules of the same module type.

LIFO

Last-in/first-out; the order in which processing is performed. For example, a LIFO queue would process data on a last-come, first-served basis. See also *FIFO*.

limit

The size or number of given items requiring system resources (such as mailboxes, locked pages, I/O requests, open files, and so on) that a job is allowed to have at any one time during execution, as specified by the system manager in the user authorization file. See also *quota*.

line

The network management component that provides a distinct physical data path.

line buffer

A storage area used to store the last line deleted by an EDT delete line operation.

line feed

The ASCII command character that moves the cursor position down one line.

line number

A number used to identify a line of text in a file processed by a text editor.

line printer

An output device that prints files one line at a time. It is used for printing large amounts of output that would otherwise tie up a slower device. Almost every system has a device designated as the line printer. In some cases, the "line printer" is actually a high-speed terminal.

link access protocol (LAP)

A set of procedures used for link control on a packet switching data network. X.25 defines two sets of procedures:

- LAP—The data terminal equipment/data circuit-terminating equipment (DTE/DCE) interface is defined as operating in two-way simultaneous asynchronous response mode (ARM) with the DTE and DCE containing a primary and secondary function.
- LAPB—The DTE/DCE interface is defined as operating in two-way asynchronous balanced mode (ABM).

linker

A system program that creates an executable program, called an image, from one or more object modules produced by a language compiler or assembler. Programs must be linked before they can be executed.

linking

The resolution of external references between the object modules used to create an image; the acquisition of referenced library routines, service entry points, and data for the image; and the assignment of virtual addresses to components of an image.

literal mode

An addressing mode in which the instruction operand is a constant value expressed in a 6-bit field of the instruction. If the operand data type is byte, word, longword, or quadword, the operand is zero-extended and can express values in the range 0 through 63 (decimal). If the operand data type is floating or double floating, the 6-bit field is composed of two 3-bit fields, one for the exponent and the other for the fraction. The operand is extended to floating or double floating format.

LMB

See *logical memory block*.

load assist agent

An image that provides additional data required to perform a downline system load to a node in a local area VAXcluster configuration.

load balancing

A function of the operating system by which work is distributed equally among all processors in a system. For more information, see *static load balancing* and *dynamic load balancing*.

load device

The drive that holds the distribution media during software installation.

local area transport (LAT)

A communications protocol that the VMS operating system uses within a local area network to communicate with terminal servers.

local area VAXcluster system

A type of VAXcluster configuration in which cluster communication is carried out over the Ethernet by software that emulates certain computer interconnect (CI) port functions. A VAXcluster node can be a VAX or a MicroVAX processor; hierarchical storage controllers (HSC) are not used.

local disk

A disk drive in a computer interconnect (CI) environment that is independent of the hierarchical storage controller (HSC).

local drive

Any drive that is connected directly to a VAX computer.

local node

The network node at which the user is physically located.

local symbol

(1) A symbol meaningful only to the module that defines it. Symbols not identified to a language processor as global symbols are considered to be local symbols. A language processor resolves (matches references with definitions) local symbols. They are not known to the linker and cannot be made available to another object module. They can, however, be passed through the linker to the symbolic debugger. Contrast with *global symbol*.

(2) A command language symbol name that is accessible only at the current command level and subsequently invoked levels. It is deleted when the command level at which it is defined exits.

locality

See *program locality*.

locate mode

A VMS Record Management Services (RMS) record access technique in which a program accesses records in a VMS RMS I/O buffer area to reduce overhead. See also *move mode*.

lock

An association between a job and a resource maintained by the VMS Lock Manager. A lock is normally used to synchronize access by multiple processes to shared objects.

lock manager

See *VMS Lock Manager*.

lock mode

A value associated with a request to the lock management system services, indicating the compatibility of the requested lock with other locks.

lock status block

Location containing the lock identification, final completion status, and optionally, a lock value block.

lock step

When two or more CPUs execute exactly the same operations at exactly the same time.

lock value block

An optional block of data associated with a lock-status block. The lock value block can be used to communicate information among processes sharing a resource.

locked password

A password that cannot be changed by the account's owner. Only system managers or users with the SYSPRV privilege can change locked passwords.

locking a page in memory

Making a page in a process ineligible for either paging or swapping. A page stays locked in physical memory until the operating system specifically unlocks it.

locking a page in the working set

Making a page within a process ineligible for paging out of the working set for the process. The page can be swapped when the process is swapped. A page stays locked in a working set until it is specifically unlocked.

log

A record of performance.

log manager

Software that provides a mechanism for storing a permanent record of transaction states in log files. A log file is a sequence of transaction records.

logging

The network management component that routes event data to logging sinks such as a console or file.

logging file

The destination to which a machine-readable record of events is sent for later retrieval. The logging file is user defined.

logging in

The identification of a user to the operating system. When users log in, they type an account name and password in response to the appropriate prompts from the system. If the name and password match an account on the system, the user will be permitted access to that account.

logging console

The destination to which a record of events is sent in a form that is comprehensible to system users. Typically, a logging console is a terminal or a user-specified file.

logging monitor

The destination to which a machine-readable record of events is sent for possible real-time decision making. Typically, the logging monitor is a user-defined program.

logging out

Entering the DIGITAL Command Language (DCL) command LOGOUT, which informs the operating system that the user has finished using a particular terminal.

logical block number (LBN)

A volume-relative address of a block on a mass storage device. The blocks that form the volume are labeled sequentially starting with logical block 0. Volume shadowing duplicates data by writing it to identical logical blocks on multiple disks. See also *physical block number* and *virtual block number*.

logical channel

A logical link between data terminal equipment (DTE) and its data circuit-terminating equipment (DCE). The physical communications line between a DTE and DCE is divided into a set of logical channels.

logical channel number (LCN)

A unique reference number that identifies a logical channel. Data terminal equipment (DTE) recognizes a virtual circuit by its associated LCN.

logical I/O function

A set of I/O operations (for example, read and write logical block) that allow restricted direct access to device level I/O operations using logical block addresses.

logical link

(1) A communication path between programs on two network nodes. Contrast with *physical link*.

(2) A carrier of a single stream of full-duplex traffic between two user-level processes.

logical memory block (LMB)

The portion of information written to the dump file during a selective crash dump. Each LMB includes a header block describing the data structure, blocks describing the parts of the address space that could not be dumped, and the dumpable parts of virtual address space.

LMBs exist for system and global page tables, system space, global pages in use by a process or processes, and for individual processes.

logical name

A user-specified name for any portion or all of a file specification. For example, the logical name INPUT can be assigned to a terminal device from which a program reads data entered by a user. Logical name assignments are maintained in logical name tables for each process, each group, and the system. Logical names can be assigned translation attributes, such as terminal and concealed. See also *search list*.

logical name table

A table that contains a set of logical names and their equivalence names for a particular process, a particular group, or the system.

logical record

A group of related fields treated as a unit.

login file

A command procedure that is automatically executed at login and at the beginning of a batch job.

longword

Four contiguous bytes (32 bits) starting on any addressable byte boundary. Bits are numbered from right to left, 0 through 31. The address of the longword is the address of the byte containing bit 0. When interpreted arithmetically, a longword is a two's complement integer with significance increasing from bit 0 to bit 30. When interpreted as a signed integer, bit 31 is the sign bit. The value of the signed integer is in the range –2,147,483,648 to 2,147,483,647. When interpreted as an unsigned integer, significance increases from bit 0 to bit 31. The value of the unsigned integer is in the range 0 to 4,294,967,295.

loop node

A local node associated with a particular line and treated as if it were a remote node. All traffic to the loop node is sent over the associated line.

loosely coupled system

A multiprocessing system configuration consisting of separate operating systems that communicate through some message transfer mechanism. Contrast with *tightly coupled system*.

macro

A statement that requests a language processor to generate a predefined set of instructions.

magnetic tape ancillary control process (MTACP)

The internal software process of the operating system that interprets the logical format of VMS ANSI-labeled volumes.

mailbox

A software data structure that is treated as a record-oriented device for general interprocess communication. Communication using a mailbox is similar to other forms of device-independent I/O. Senders write to a mailbox, the receiver reads from that mailbox. Some systemwide mailboxes are defined; the error logger and operator communication manager (OPCOM) read from systemwide mailboxes.

main memory

See *physical memory*.

main text buffer

In a text editor, the default text buffer for keyboard input and for input files, and the source for output files.

manual record unlocking

A VMS Record Management Services (RMS) capability that allows users to lock multiple records in a file simultaneously. The user has explicit control over the unlocking of records. A lock occurs when the RAB$V_ULK bit is set in the record processing options field on the execution of a Get, Find, or Put service. Once a record is locked when record unlocking is enabled, it will remain locked until it is explicitly unlocked by either the Free or Release service, or until the stream terminates.

mapping window

A subset of file retrieval information used to translate virtual block numbers to logical block numbers.

marginal vector consumer

A process executing an image that has issued VAX vector instructions, but has not issued any instructions in the amount of time determined by the VECTOR_MARGIN system parameter. A marginal vector consumer relinquishes its need for the VECTOR capability and, thus, need no longer be scheduled on an available scalar-vector processor pair. A marginal vector consumer is eligible for execution on any processor in the system. See *vector consumer*.

mass storage device

An input/output device on which data and other types of files are stored while they are not being used. Typical mass-storage devices include disks, magnetic tapes, and floppy disks.

mass storage control protocol (MSCP)

The software protocol used to communicate I/O commands between a VAX processor and DSA-compliant devices on the system.

MASSBUS adapter (MBA)

An interface device between the backplane interconnect and the MASSBUS.

master file directory (MFD)

The file directory on a disk volume that contains the name of all user file directories (UFDs) on a disk, including its own.

maximum visits

The maximum number of nodes through which a packet can be routed before reaching its destination.

MBA

See *MASSBUS adapter*.

MBZ

See *must be zero*.

MCR

See *monitor console routine*.

mean-time-between-failures (MTBF)

The average time that passes before a computing component fails such that remedial action is required.

member number

The second number or its alphanumeric equivalent in a user identification code (UIC) that uniquely identifies that code.

member unit

See *shadow set member*.

memory

A series of physical locations into which data or instructions can be placed in the form of binary words. Each location in memory can be addressed and its contents can be altered. Memory should not be confused with mass-storage devices.

memory management

The operating system functions that include the hardware's page mapping and protection and the operating system's image activator and pager.

memory mapping enable (MME)

A bit in a processor register that governs address translation.

merge member

A shadow set member that is partially consistent with all other merge members of the shadow set. Only partial inconsistencies exist among the merge members so that some number of blocks were written and some were not at the time of a hardware failure.

merge operation

An operation that brings shadow set members, which are in a partially consistent state, into a fully consistent state while also permitting I/O operations to occur on the shadow set. A merge operation also attempts to correct blocks that are inconsistent on disk devices that were once members of the same shadow set. For a merge operation to occur, the shadowing software recognizes that the shadow set members are all former members of the same shadow set and the data on one member is just as correct as the data on any other member.

MFD

See *master file directory*.

mixed-interconnect VAXcluster system

A VAXcluster configuration that uses both the computer interconnect (CI) and Ethernet for interprocessor communication. Supported in VAXcluster systems running VMS Version 5.0 or higher.

MME

See *memory mapping enable*.

modem

Contraction of modulator/demodulator. A device that modulates signals for sending over communications facilities and demodulates signals being received.

modify access type

A type of access in which the specified operand of an instruction or procedure is read and potentially modified and written during execution.

module

(1) A portion of a program or program library, as in a source module, object module, or image module.

(2) A board, usually made of plastic covered with an electrical conductor, on which logic devices (such as transistors, resistors, and memory chips) are mounted, and circuits connecting these devices are etched, as in a logic module.

(3) A network management component.

monitor console routine (MCR)

The command interpreter in an RSX–11 system; also an optional command interpreter in a VMS system.

mounting a volume

(1) Logically associating a volume with the physical unit on which it is loaded (an activity accomplished by system software at the request of an operator).

(2) Loading or placing a magnetic tape or disk pack on a drive and placing the drive on line (an activity accomplished by a system operator).

mount verification

A VMS feature that suspends I/O to and from volumes while they are changing status. Mount verification also ensures that, following a suspension in disk I/O, the volume being accessed is the same as was previously mounted. When volume shadowing is installed, the mount verification feature also maintains and validates shadow set membership for Files–11 shadow sets.

move mode

A VMS Record Management Services (RMS) record I/O access technique in which a program accesses records in its own working storage area. See also *locate mode*.

MSCP

See *mass storage control protocol*.

MTACP

See *magnetic tape ancillary control process*.

MTBF

See *mean-time-between-failures*.

multiaccess channel

A media (for example, Ethernet) on which many transmitters contend for access.

multicast addressing

An Ethernet addressing mode in which a given message packet is targeted to a group of logically related nodes.

multicast group address

An address assigned to a number of nodes on an Ethernet and used to send a message to all nodes in the group in a single transmission.

multinational character set

A set of 8-bit character international alphanumeric characters, including characters with diacritical marks.

multipoint circuit

A circuit connecting two systems, with one of the systems (the control station) controlling the circuit and the other system serving as a tributary.

multiport memory

A memory unit that can be connected to multiple processors and that can contain resources (for example, mailboxes, common event flag clusters, and global sections) for use by processes running on different processors.

multiprocessing system

A system containing two or more general-purpose processors. These processors are connected, through hardware, so that they can work on the same application concurrently. See *asymmetric multiprocessing* and *symmetric multiprocessing*.

multiprogramming

A mode of operation in which hardware resources are shared among multiple, independent software processes.

must be zero (MBZ)

A field that is reserved and must be supplied as zero. If examined, it must be assumed to be undefined.

mutex

A semaphore used to control exclusive access to a region of code that can share a data structure or other resource. The mutex (mutual exclusion) semaphore ensures that only one process at a time has write access to the region of code.

NAM

See *name block*.

name block (NAM)

A VMS Record Management Services (RMS) user data structure that contains supplementary information used in parsing file specifications.

National Replacement Character (NRC)

A 7-bit character set that is a country-specific version of ASCII.

native image

An image whose instructions are executed in native mode.

native mode

The processor's primary execution mode, where the programmed instructions are interpreted as byte-aligned, variable-length instructions that operate on four data types: byte, word, longword, and quadword integers; floating and double floating character strings; packed decimals; and variable-length bit fields. The other instruction execution mode is compatibility mode.

NCP

See *Network Control Program.*

NETACP

See *network ancillary control process.*

network

A collection of interconnected individual computer systems.

network ancillary control process (NETACP)

A VMS ancillary control process that controls all lines and circuits, maintains a picture of the network topology, and creates a process to receive inbound logical link requests.

network connect block (NCB)

(1) For a DECnet network, a user-generated data structure used in a nontransparent task to identify a remote task and optionally send user data in calls to request, accept, or reject a logical link connection.

(2) For the VAX Packetnet System Interface (PSI), a block that contains the information necessary to set up an X.25 virtual circuit or to accept or reject a request to set up an X.25 virtual circuit.

Network Control Program (NCP)

An interactive utility program that permits you to control and monitor the network.

network object

Any task with a nonzero object type, for example, those programs such as FAL and NML that provide generic services across a network.

network services protocol (NSP)

A formal set of conventions used in a DECnet network to perform network management and to exchange messages over logical links.

network status notification

Information about the state of both logical and physical links over which two tasks communicate. A nontransparent task can use this information to take appropriate action under conditions such as third-party disconnections and a partner's exiting before I/O completion.

network task

A nontransparent task that is able to process multiple inbound connection requests; that is, it has declared a network name or object number.

node

(1) An individual computer system in a network that can communicate with other computer systems in the network.

(2) A VAXBI interface—such as a central processor, controller, or memory subsystem—that occupies one of 16 logical locations on a VAXBI bus. See also *VAXBI*.

(3) A VAX processor or HSC that is recognized by system communications services (SCS) software.

node address

The required, unique, numeric identification of a specific node in the network.

node name

An optional alphanumeric identification associated with a node address in a strict one-to-one mapping. A node name must contain at least one alphabetic character.

node specification

The first field in a file specification. This field identifies the location of a computer system in a network.

nonprinting character

A character in the computer code set for which there is no corresponding graphic symbol.

nonprivileged

In DECnet VAX terminology, this term means no privileges in addition to NETMBX, which is the minimal requirement for any network activity.

nonrouting node

See *end node*.

NRC

See *National Replacement Character*.

NSP

See *network services protocol*.

null

(1) The character with the ASCII code 000.

(2) An absence of information.

null string

A string without content or an empty string represented by adjacent quotation marks.

numeric string

A contiguous sequence of bytes representing up to 31 decimal digits (one per byte) and possibly a sign. The numeric string is specified by its lowest addressed location, its length, and its sign representation.

object

(1) A DECnet VAX process that receives a logical link request. It performs a specific network function or is a user-defined image for a special-purpose application.

(2) A VAX Packetnet System Interface (PSI) management component that contains records to specify account information for incoming calls and to specify a command procedure that is initiated when the incoming call arrives.

(3) A system resource such as a file, device, or directory.

object module

The binary output of a language processor such as the assembler or a compiler, which is used as input to the linker.

object program

A source program that has been translated into machine language.

object time system

See *Run-Time Procedure Library*.

object type

A discrete identifier for either a task or DECnet service on a remote node. Object type identifiers can either be 0 plus a name (alternatively, TASK=name) or nonzero without a name (for example, 17= or FAL=).

octal number

A number in the base-8 numbering system. Only the numerals 0 through 7 are used in this system. If a number includes an 8 or a 9, it cannot be an octal number. Octal numbering is used in computer systems because it is easy to convert to the binary numbers that are actually used by the computer.

offset

A fixed displacement from the beginning of a data structure. System offsets for items within a data structure normally have an associated symbolic name used instead of the numeric displacement. Where symbols are defined, programmers always reference the symbolic names for items in a data structure instead of using the numeric displacement.

OLTP

See *Online Transaction Processing*.

On-Disk Structure Level 1 (ODS1)

See *Files–11 On-Disk Structure Level 1*.

On-Disk Structure Level 2 (ODS2)

See *Files–11 On-Disk Structure Level 2*.

Online Transaction Processing (OLTP)

An environment in which many users are simultaneously reading and writing to a collection of shared data, generally a database. Results are usually expected immediately (real time). In contrast to standard transaction processing, when an OLTP transaction is completed, all data is fully updated during the request and output stages. No files are created for later batch updates of the stored data. See also *transaction processing*.

opcode

The pattern of bits within an instruction that specifies the operation to be performed.

OPCOM

See *operator communication manager*.

open account

An account that does not require a password.

operand specifier

The pattern of bits in an instruction that indicates the addressing mode and register, or a displacement that identifies an instruction operand.

operand specifier type

The access type and data type of an instruction's operand or operands. For example, the test instructions are of read-access type because they only read the value of the operand. The operand can be of byte, word, or longword data type, depending on whether the opcode is for the TSTB (test byte), TSTW (test word), or TSTL (test longword) instruction.

operating system

An integrated collection of programs that controls the execution of computer programs and performs system functions.

operator

The person responsible for daily maintenance of the system at a particular installation. The operator does such things as mounting disks and tapes, changing ribbons and paper on line printers, rebooting the system, keeping records, and so forth. In small systems, these duties may be combined with those of the system manager or informally divided among several people.

operator communication manager (OPCOM)

A system process that receives input from a process that wants to inform an operator of a particular status or condition, passes a message to the operator, and tracks the message.

operator's console

Any terminal identified as a terminal attended by a system operator.

out-of-order packet caching

The mechanism by which the network services protocol (NSP) maintains a buffer of data packets received out of order so that they can be reassembled in the correct order before being forwarded to the destination node.

outbound connection

A task's request for a logical link connection to another node.

output file

A file that contains the results of a processing operation, for example, a file that has been sorted or edited.

OWNER

In the context SYSTEM/OWNER/GROUP/WORLD, an owner is the particular member (of a GROUP) to which a file, global section, mailbox, or event flag cluster belongs.

owner process

The process or subprocess that created a subprocess.

P0

See *program region*.

P0BR

See *program region base register*.

P0LR

See *program region length register*.

P0PT

See *program region page table*.

P1

See *control region*.

P1 through P8

See *parameter*.

P1BR

See *control region base register*.

P1LR

See *control region length register*.

P1PT

See *control region page table*.

packed decimal

A method of representing a decimal number by storing a pair of decimal digits in 1 byte, taking advantage of the fact that only four bits are required to represent the numbers 0 through 9.

packed decimal string

A contiguous sequence of up to 16 bytes interpreted as a string of 4-bit fields. Each field represents a digit except for the low-order four bits of the highest addressed byte, which represent the sign. The packed decimal string is specified by its lowest addressed location and by the number of digits.

packet

A unit of data to be routed from a source node to a destination node; for the VAX Packetnet System Interface (PSI), the unit of data switched through a packet switching data network (PSDN). Normally, a user data field accompanied by a header carrying destination and other information.

packet assembly/disassembly (PAD) facility

A device at a packet switching data network (PSDN) that allows access from an asynchronous terminal. The terminal connects to the PAD, and the PAD puts the terminal's input data into packets (assembles) and takes the terminal's output data out of packets (disassembles).

packet switching

A data transmission process, utilizing addressed packets, whereby a channel is occupied only for the duration of transmission of the packet.

Packetnet System Interface (PSI)

A software product that allows the VMS user to communicate across packet switching data networks (PSDNs).

PAD

See *packet assembly/disassembly (PAD) facility*.

packet switching data network (PSDN)

A set of equipment and interconnecting links that provides a packet switching communications service to subscribers.

page

(1) A set of 512 contiguous byte locations beginning at an even 512-byte boundary used as the unit of memory mapping and protection.

(2) The data between the beginning of file and a page marker, between two markers, or between a marker and the end of a file.

page fault

An exception generated by a reference to a page that is not in the working set of the faulting process. Also called *translation-not-valid fault*.

page fault cluster

See *cluster*.

page fault cluster size

The number of pages read into memory on a page fault.

page frame number (PFN)

The high-order 21 bits of the physical address of a page in physical memory.

page frame number mapping (PFN mapping)

Mapping a section to one or more pages in physical memory or I/O space (as opposed to mapping it to a disk file).

page marker

A character or characters (generally a form feed) that separates pages in a file processed by a text editor.

page table

A memory management data base used to account for virtual pages. See also *system page table, process page table,* and *global page table*.

page table entry (PTE)

The data structure that identifies the physical location and status of a page of virtual address space. When a virtual page is in memory, the PTE contains the page frame number needed to map the virtual page to a physical page. When it is not in memory, the page table entry contains the information needed to locate the page on secondary storage (disk).

pager

A set of kernel mode procedures that executes as the result of a page fault. The pager makes the page for which the fault occurred available in physical memory so that the image can continue execution. The pager and the image activator provide the operating system's memory management functions.

paging

The action of bringing pages of an executing process into physical memory when referenced. When a process executes, all of its pages are said to reside in virtual memory. Only the actively used pages, however, need to reside in physical memory. The remaining pages can reside on disk until they are needed in physical memory. On a VMS system, a process is paged either when it references more pages than it is allowed to have in its working set or when it first activates an image in memory. When the process refers to a page not in its working set, a page fault occurs. This faulting causes the operating system's pager to read in the referenced page if it is on disk (and, optionally, other related pages depending on a cluster factor), replacing the least recently faulted pages as needed. The operating system's pager does not read in a referenced page if that page is on the free or modified list.

parallel processing

A method of computing that occurs when a section of an application is divided into multiple tasks, and those multiple tasks are executed simultaneously on multiple processors.

parameter

(1) A value passed to a command procedure equated to a symbol ranging from P1 through P8. See also *command parameter*.

(2) An entry in the volatile or permanent database for a network management component.

parsing

(1) Breaking a command string into its elements to interpret it.

(2) Interpreting a file specification, as is done by VMS Record Management Services (RMS).

password

A character string that users provide at login time to validate their identity and as a form of proof of their authorization to access the account. There are two kinds of passwords—system passwords and user passwords. User passwords include both primary and secondary passwords.

path

The route a packet takes from source to destination.

path cost

The sum of the circuit costs along a path between two nodes.

path length

The number of hops along a path between two nodes; that is, the number of circuits a packet must travel along to reach its destination.

PC

See *program counter*.

PCB

See *process control block*.

PCBB

See *process control block base register*.

peripheral device

Any unit, distinct from the CPU and physical memory, that can provide the system with input or accept any output from it. Terminals, line printers, and disks are peripheral devices.

permanent database

A file containing information about network management components. See also *volatile database* and *configuration database*.

permanent virtual circuit (PVC)

A permanent logical association between two DTEs (data terminal equipment), which is analogous to a leased line. Packets are routed directly by the network from one DTE to the other.

per-process address space

See *process address space*.

PFN

See *page frame number*.

PFN mapping

See *page frame number mapping*.

Phase II VAXcluster configuration

A VAXcluster environment in which both the computer interconnect (CI) and the Ethernet are used for cluster communications.

physical address

(1) The address used by hardware to identify a location in physical memory or on directly addressable secondary storage devices such as disk. A physical memory address consists of a page frame number and the number of a byte within the page. A physical disk block address consists of a cylinder or track and sector number.

(2) The unique address value associated with a given system on an Ethernet circuit. An Ethernet physical address is defined to be distinct from all other physical addresses on an Ethernet.

physical address space

The set of all possible physical addresses that can be used to refer to locations in memory (memory space) or device registers (I/O space).

physical block number

The physical address of a block on a mass storage device. Contrast with *logical block number* and *virtual block number*.

physical I/O functions

A set of I/O functions that allows access to all device-level I/O operations except maintenance mode.

physical link

A signal-carrying media that links two nodes in a network. Contrast with *logical link*.

physical memory

The memory modules connected to the backplane interconnect that are used to store both instructions that the processor can directly fetch and execute and any other data that a processor is instructed to manipulate. Also called main memory.

PID

See *process identification*.

point-to-point circuit

A circuit that connects two nodes, operating over a single line.

polling

The activity that the control station performs with a multipoint circuit's tributaries to grant the tributaries permission to transmit.

position-dependent code

Code that can execute properly only in the locations in virtual address space that are assigned to it by the linker.

position-independent code

Code that can execute properly without modification wherever it is located in virtual address space, even if its location is changed after it is linked. Generally, this code uses addressing modes that form an effective address relative to the program counter (PC).

primary key

The mandatory key within the data records of an indexed file; used by VMS Record Management Services (RMS) to determine the placement of records within the file and to build the primary index. See *key (indexed files)* and *alternate key*.

primary password

A type of user password that is the first password requested from the user. Systems may optionally require a secondary password, as well. As a user password, this password must be associated with the user name that is supplied with it.

primary processor

The processor in a VMS symmetric multiprocessing system that is either logically or physically attached to the console device. Only the primary processor performs the initialization activities that define the VMS environment and prepare memory for the entire system. In addition, the primary processor serves as the system timekeeper. Contrast with *secondary processor*.

primary vector

A location that contains the starting address of a condition handler to be executed when an exception condition occurs. If a primary vector is declared, that condition handler is the first handler to be executed.

priority

See *process priority* and *interrupt priority level*.

private section

An image section of a process that is not shareable among processes. See also *global section*.

private volume

A volume that has been allocated by a process for its own exclusive use.

privilege

A means of protecting the use of certain system functions that can affect system resources and integrity. System managers grant privileges according to user's needs and deny them to users as a means of restricting their access to the system. See also *process privileges*, *user privileges*, and *image privileges*.

privileged

Generally refers to instructions, images, or accounts intended for use by the operating system, specific system programs, or a subset of the system users.

procedure

A routine entered by means of a call instruction. See also *command procedure*.

process

The basic entity scheduled by the system software, a process provides the context in which an image executes. A process consists of an address space and both hardware and software context.

process address space

See *process space*.

process affinity

In a VMS symmetric multiprocessing system, a close association of a process with a specific processor or set of processors in the system. Process affinity can be indicated as either a requirement that a process run only on the processor with a specific CPU identification or on a processor or set of processors that have a needed capability. See *capability*.

process context

The set of data defining the environment, both hardware and software, in which a process executes. See also *hardware context* and *software context*.

process control block (PCB)

A data structure used to contain process context. The hardware PCB contains the hardware context. The software PCB contains the software context, which includes a pointer to the hardware PCB.

process control block base register (PCBB)

A processor register that contains the physical address of the current process control block (PCB).

process header

A data structure that contains the hardware process control block (PCB), accounting and quota information, process section table, working set list, and the page tables defining the virtual layout of the process.

process header slots

That portion of the system address space in which the system stores the process headers for the processes in the balance set. The number of process header slots in the system determines the number of processes that can be in the balance set at any one time.

process identification (PID)

A 32-bit binary value that uniquely identifies a process. Each process has a process identification and a process name.

process I/O channel

See *channel*.

process I/O segment

That portion of a process control region that contains the process permanent VMS Record Management Services (RMS) internal file access block for each open file, and the I/O buffers, including the command interpreter's command buffer and command descriptors.

process name

A 1- to 15-character ASCII string that can be used to identify processes executing under the same group number.

process page tables

The page tables used to describe process virtual memory.

process permanent file

A file that is opened or created through VMS Record Management Services (RMS) by supervisor or executive mode when the process permanent bit is set in the file-processing options field.

process priority

The priority assigned to a process for scheduling purposes. The operating system recognizes 32 levels of process priority, where 0 is low and 31 high. Levels 16 through 31 are used for real-time processes. The system does not modify the priority of a real-time process (although the system manager or the process itself may). Levels 0 through 15 are used for normal processes. The system may temporarily increase the priority of a normal process based on the activity of the process.

process privileges

The privileges granted to a process by the system. These privileges are a combination of user privileges and image privileges. They include, for example, the privilege to affect other processes associated with the same group as the user's group, to affect any process in the system regardless of user identification code (UIC), to set process swap mode, to create permanent event flag clusters, to create another process, to create a mailbox, to perform direct I/O to a file-structured device, and to perform network operations.

process section

See *private section*.

process space

The lowest addressed half of virtual address space, where process instructions and data reside. Process space is divided into a program region and a control region.

processor register

A part of the processor used by the operating system software to control the execution states of the computer system. Processor registers include, for example, the system base and length registers, the program and control region base and length registers, the system control block base register, and the software interrupt request register.

processor status longword (PSL)

A privileged processor register consisting of a word of privileged processor status and the processor status word itself. The privileged processor status information includes the current interrupt priority level, the previous access mode, the current access mode, the interrupt stack bit, the trace trap pending bit, and the compatibility mode bit.

processor status word (PSW)

The low-order word of the processor status longword. Processor status information includes the condition codes (carry, overflow, 0, negative), the arithmetic trap enable bits (integer overflow, decimal overflow, floating underflow), and the trace enable bit.

program

A series of instructions aimed at a particular result. Programming languages are a means of describing procedures so that they can be performed by a computer. See also *image*.

program counter (PC)

General register 15 (R15). At the beginning of an instruction's execution, the PC normally contains the address of a location in memory from which the processor will fetch the next instruction it will execute.

program locality

A characteristic of a program that indicates how close or far apart the references to locations in virtual memory are over time. A program with a high degree of locality does not refer to many widely scattered virtual addresses in a short period of time.

program region

The lowest addressed half of process address space (P0 region). The program region contains the image currently being executed by the process and other user code called by the image.

program region base register (P0BR)

The processor register, or its equivalent in a hardware process control block, that contains the base virtual address of the page table entry for virtual page number 0 in a process program region.

program region page table (P0PT)

The page table that maps the program region of virtual address space.

program region length register (P0LR)

The processor register, or its equivalent in a hardware process control block, that contains the number of entries in the page table for a process program region.

program section (PSECT)

A portion of a program with a given protection and set of storage management attributes. Program sections that have the same attributes are gathered together by the linker to form an image section.

programmer number

See *member number*.

project number

See *group number* or *account number*.

prompt

A character string appearing on a terminal indicating that the user must provide input.

protection

The attributes of a resource that limit the type of access available to users. Resources include volumes, devices, directories, and files. See also *user identification code* and *access control list*.

protocol

An agreed set of rules governing the operation of a communications link.

proxy login

The procedure that permits a remote user to access a specific account at the local node, without supplying the user name and password.

PSECT

See *program section*.

pseudodevice

An entity treated as an I/O device by the user or system, although it is not any particular physical device. A pseudodevice is a forwarding address through which actual physical devices can always be reached.

PSDN

See *packet switching data network*.

PSI

See *Packetnet System Interface*.

PSL

See *processor status longword*.

PSW

See *processor status word.*

PTE

See *page table entry.*

public volume

A file-structured disk volume that contains public files.

PVC

See *permanent virtual circuit.*

pure code

See *reentrant code.*

QIO

See *Queue I/O Request system service.*

quadword

Four contiguous words (64 bits) starting on any addressable byte boundary. Bits are numbered 0 to 63 from right to left. A quadword is identified by the address of the word containing the low-order bit (bit 0). When interpreted arithmetically, a quadword is a two's complement integer with significance increasing from bit 0 to bit 62. Bit 63 is used as the sign bit. The value of the integer is in the range -2^{63} to $2^{63}-1$.

qualifier

A portion of a command string that modifies a command verb or command parameter by selecting one of several options. A qualifier, if present, follows the command verb or parameter to which it applies and is in the format /qualifier[=option]. For example, in the command string "PRINT filename /COPIES=3," the COPIES qualifier indicates that the user wants three copies of a given file printed.

quantum

The minimum amount of time that a process can remain in memory; also the maximum amount of time that a process can be the executing process. A specified amount is deducted from the quantum whenever a process enters a wait state.

queue

(1) A line of jobs to be processed, for example, a batch job queue or a printer job queue. Processing occurs primarily in first-in/first-out (FIFO) order, but does reflect the priority of the process that submitted the job. See also *state queue* and *system queue.*

(2) To make an entry in a list or table, perhaps using the INSQUE instruction.

Queue I/O Request system service

The VMS system service that handles $QIO and $QIOW requests. The QIO prepares an I/O request for processing by the driver and performs device-independent preprocessing of the request. This system service also calls driver function decision table (FDT) routines.

queue priority

The priority assigned to a job placed in a spooler queue or a batch queue.

quota

The total amount of a system resource, such as CPU time, that a job is allowed to use in an accounting period, as specified by the system manager in the user authorization file. See also *limit*.

RAB

See *record access block*.

random access

A method for retrieving or writing data in which the location of the data to be retrieved or written is not dependent on the location of previously retrieved or written data. Random access refers to memory or mass-storage devices on which all information is equally accessible.

random access by key

The retrieval or storage of a record by specifying the key value. This method of record retrieval and storage applies only to indexed files.

random access by record file address (RFA)

The retrieval of a record by its unique address, which is provided to the program by VMS Record Management Services (RMS) upon successful $GET or $FIND operations. The record's file address can subsequently be used to randomly access that same record.

random access by relative record number

The retrieval or storage of a record by specifying its position relative to the beginning of the file. This method of record storage and retrieval applies only to sequential files with fixed-length records and relative files.

reachable node

A node to which the local node has a usable communications path.

read

The act or capability of an image to accept data. For example, when a TYPE command is entered, the system reads the designated file from the disk and writes it to the terminal. See also *write*.

read access type

An instruction or procedure operand attribute indicating that the specified operand is only read during instruction or procedure execution.

read miss

An event in which a read operation cannot be serviced by the processor's hardware cache.

real-time process

A process that responds to events in related or controlled processes as they occur, rather than when the computer is ready to respond to them. The results of the computation can thus be used in the processing. A real-time process is assigned to a software priority level between 16 and 31, inclusive. The scheduling priority assigned to a real-time process is never modified by the scheduler, although it can be modified by the system manager or by the process itself.

record

A set of related data that a program treats as a unit.

record access block (RAB)

A VMS Record Management Services (RMS) user control block allocated at either assembly or run time to communicate with VMS RMS. The control block describes the records in a particular file and associates with a file access block to form a record access stream. A RAB defines the characteristics needed to perform record-related operations, such as update, delete, or get.

record access mode

The method used in VMS Record Management Services (RMS) for retrieving and storing records in a file. Access is by one of four methods: sequential, random by key, random by record's file address, and random by relative record number.

record access mode switching

Term applied to the switching from one type of record access mode to another while processing a file.

record blocking

The technique of grouping multiple records into a single block. On magnetic tape, an interrecord gap (IRG) is placed after the block rather than after each record. This technique reduces the number of I/O transfers required to read or write the data; and, in addition (for magnetic tape), increases the amount of usable storage area. Record blocking also applies to disk files.

record cell

A fixed-length area in a relative file that is used to contain one record.

record format

The way a record physically appears on the recording surface of the storage media. The record format defines the method for determining record length.

record length

The size of a record in bytes.

record locking

The ability to control operations being performed on relative and indexed files that are being simultaneously accessed by more than one program or more than one record stream. Record locking makes certain that when a program is adding, deleting, or modifying a record on a given stream, another program or stream is not allowed to access the same record or record cell. See also *automatic record locking* and *manual record locking*.

Record Management Services (RMS)

A set of operating system procedures that is called by programs to process files and records within files. VMS RMS allows programs to issue GET and PUT requests at the record level (record I/O) as well as read and write blocks (block I/O). VMS RMS is an integral part of the system software; its procedures run in executive mode.

record-oriented device

A device such as a terminal, line printer, or card reader on which the largest unit of data a program can access in one I/O operation is the device's physical record.

record file address (RFA)

The unique address of a record in a file, the RFA allows previously accessed records to be accessed randomly at a subsequent time. This access occurs regardless of file organization.

recoverability

The ability of a system to reconfigure itself and continue (or quickly resume) operation.

redundant

Duplicate or extra computing components that protect a computing system from failure.

reentrant code

Code that is never modified during execution. It is possible to let many users share the same copy of a procedure or program written as reentrant code.

reentrant service

A service that is safe to call from multiple threads in parallel. If a service is reentrant, there is no burden placed on calling routines to serialize their access or take other explicit precautions. See also *thread-serial service* and *thread-synchronous service*.

register

A storage location in hardware logic other than main memory. See also *general register, processor register,* and *device register*.

register deferred indexed mode

An indexed addressing mode in which the base operand specifier uses register deferred mode addressing. See also *indexed addressing mode* and *register deferred mode*.

register deferred mode

An addressing mode in which the contents of the specified register are used as the address of the actual instruction operand. See also *addressing mode*.

register mode

An addressing mode in which the contents of the specified register are used as the actual instruction operand.

relative file organization

The arrangement of records in a file in which each record occupies a cell of equal length within a bucket. Each cell is assigned a successive number, which represents its position relative to the beginning of the file.

relative record number

An identification number used to specify the position of a record cell relative to the beginning of the file. The relative record number is used as the key during random access by key mode to relative files. To randomly access a record in a sequential disk file having 512-byte, fixed-length records, a program must provide VMS Record Management Services (RMS) with the relative record number of the cell containing the record.

relative volume number

A volume number that uniquely identifies which volume set contains a file.

reliability

The ability of a computing system to operate without failing. Reliability is measured by a *Mean Time Between Failure (MTBF)* formula.

remote device

A device that is not directly connected to the local node, but is available through the VAXcluster system.

remote data terminal equipment (DTE)

Any DTE in a network other than the one at which the user is located.

remote node

To any one node in the network, this node is any other network node. See also *adjacent node*, *local node*, and *executor node*.

reorganization

A record-by-record copy of an indexed file to another indexed file with the same key attributes as the input file.

resource

(1) A physical part of the computer system such as a device or memory, or an interlocked data structure such as a mutex. Quotas and limits control the use of physical resources.

(2) Any entity to which access is synchronized by means of the lock management system services.

resource manager

Software that participates in distributed transactions and manages shared access to a recoverable resource. An example of a recoverable resource is a database system.

resource wait mode

An execution state in which a process indicates, when it issues a service request requiring a resource, that it will wait until a system resource becomes available. If a process requests notification when a resource is not available, it can disable resource wait mode during program execution.

resume

To activate a suspended process. Contrast with *wake*.

return status code

See *status code*.

reverse video

A feature of a video terminal that reverses the default video contrast. If the default display is black figures on a white background, reverse video displays white figures on a black background.

RFA

See *record file address*.

rights database

The collection of data the system maintains and uses to define identifiers and associate identifiers with the holders of the identifiers.

rights list

The list associated with each process that includes all the identifiers the process holds.

RMS

See *Record Management Services*.

RMS–11

A set of routines that are linked with compatibility mode programs and provide similar functional capabilities to VMS Record Management Services (RMS). The file organizations and record formats used by RMS–11 are very similar to those of VMS RMS.

round robin

A form of timesharing that gives images of equal priority equal access to the CPU. The VMS operating system uses round-robin scheduling for each of the lower 16 software priority levels. Each process at a given software priority level executes in turn before any other process at that level (a first-in/first-out (FIFO) queue).

router

A node that can send, receive, and route packets from one node to another.

routing

The network function that determines the path along which data travels to its destination.

runaway tape condition

A situation where a tape spins unceasingly on the drive. A runaway tape condition usually occurs because an operation does not incur a timeout condition. The only way to recover from a runaway tape condition is to take the drive off line.

Run-Time Procedure Library

The collection of procedures available to native-mode images at run time. These procedures may be used by all native-mode images, regardless of the language processor used to compile or assemble the program. These procedures also provide support routines for high-level language compilers.

RVN

See *relative volume number*.

RWED

The abbreviation for Read, Write, Execute, Delete, which are types of access to data files, directory files, or volumes.

save set

A file that the Backup Utility creates to save files that it backs up.

satellite node

A VAX processor that is part of a local area VAXcluster system. A satellite node is booted remotely from the system disk of the boot server in this type of VAXcluster system. See also *boot server*.

SBI

See *synchronous backplane interconnect*.

SBR

See *system base register*.

scalar

A single data item, having one value.

scalar consumer

A process executing an image that issues VAX scalar instructions only. Contrast with *vector consumer*.

scaleability

How well the software or hardware product is able to adapt to future business needs.

scatter/gather

The ability to transfer in one I/O operation data from discontiguous pages in memory to contiguous blocks on disk, or data from contiguous blocks on disk to discontiguous pages in memory.

scavenging

See *disk scavenging*.

SCB

See *system control block*.

SCBB

See *system control block base register*.

scheduling priority

See *process priority*.

screen width

The number of character positions that can be displayed on a line.

scrolling

A feature of a video terminal that allows the display of more than one screenful of text by vertical movement. For example, when the TYPE command is entered, new output appears at the bottom of the screen as the oldest output disappears off the top.

SCS

See *system communications services*.

search list

A logical name in which the equivalence name has multiple values instead of a single value. A common use of a search list is to examine multiple directories to locate a file.

search string

A group of characters defined in a command as the object of a search operation.

secondary password

A user password that may be required at login time, immediately after the primary password has been correctly submitted. Primary and secondary passwords can be known by separate users, to ensure that more than one user is present at the login. A less common use is to require a secondary password as a means of increasing the password length so that the total number of combinations of characters makes infiltrating a system by password more time-consuming.

secondary processor

The processor or processors in a VMS symmetric multiprocessing system that do not have the initialization and timekeeper responsibilities of the primary processor.

secondary storage

Random-access mass storage that is implemented on disks.

secondary vector

A location that identifies the starting address of a condition handler to be executed when a condition occurs and when either the primary vector contains 0 or the handler to which the primary vector points chooses not to handle the condition.

section

A portion of process virtual memory that has common memory management attributes (protection, access, cluster factor, and so on). It is created from an image section, a disk file, or as the result of a Create Virtual Address Space system service. See *global section*, *private section*, *image section*, and *program section*.

secure terminal server

A piece of VMS software designed to ensure that users can only log in to terminals that are already logged out. When the user presses the Break key on a terminal, the secure server (if enabled) responds by first disconnecting any logged in process and then initiating a login. If no process is logged in at the terminal, the login can proceed immediately.

security alarm

A message sent to operator terminals that are enabled as security operators. Security alarms are triggered by the occurrence of an event previously designated as worthy of the alarm because of its security implications.

security operator terminal

A class of terminal that has been enabled to receive messages sent by the operator communication manager (OPCOM) to "security operators." These messages are security alarm messages. Normally such a terminal is a hardcopy terminal in a protected room, so that the output provides a log of security-related events and details that identify the source of the event.

security auditing

The monitoring and recording of specified events occurring on the system. These events include login failures, privileged and unprivileged access to system objects, and changes to the system user authorization file (SYSUAF).

selective crash dump

A crash dump that saves only those portions of physical memory critical to an analysis of system failure.

semaphore

In a DECnet network, a common data structure used to control the exchange of signals between concurrent processes.

sequential access mode

The retrieval or storage of records where a program reads or writes records one after the other in the order in which they appear, starting and ending at any arbitrary point in the file.

sequential file organization

A file organization in which records appear in the order in which they were originally written. The records can be fixed length or variable length. Sequential file organization permits sequential record access and random access by the record's file address. Sequential file organization with fixed-length records also permits random access by relative record number.

served device

A device whose local node makes it available to other nodes in the VAXcluster system.

server

A computing system entity that provides a service to other system entities called *clients*.

shadow set

One or more disk volumes of the same device type and the same physical LBN geometry that are united for volume shadowing and represented by a virtual unit. Thus, the term shadow set refers to the physical units *and* the virtual unit.

shadow set member

A physical disk mounted as part of a shadow set. Normally, shadow set members are consistent with each other both structurally and in content.

shareable image

An image that has all of its internal references resolved, but must be linked with one or more object modules to produce an executable image. A shareable image cannot be executed. A shareable image file can be used to contain a library of routines. A shareable image can be used to create a global section by the system manager.

shared image

An image that is installed so that multiple users in a system can share the memory pages where the image is loaded.

shared memory

A generic term referring to any memory that can be accessed by two or more concurrent processes. In a VMS symmetric multiprocessing system, a single copy of the VMS operating system resides in memory. Each processor in the system can access this memory, as can any process executing on any processor. See also *multiport memory*.

shell process

A predefined process that the job controller copies to create the minimum context necessary to establish a process.

shrinking the working set

An alternative available to the swapper process to obtain pages in physical memory. The swapper will shrink the size of the working set of selected processes to obtain pages in physical memory. See also *swapping*.

signal

(1) An electrical impulse conveying information.

(2) The software mechanism used to indicate that an exception condition was detected.

(3) In threads, a mechanism used to wake only one thread waiting for a condition variable. See also *broadcast*.

single point of failure

A portion of a computing system that, if it fails, causes the system to cease providing service.

sink node

A node on which logging sink types, such as a file or console, are actually located.

slave terminal

A terminal that sends and receives I/O from an image and directly from the operating system. It is not possible to enter commands to the command interpreter from a slave terminal.

SLR

See *system length register*.

small process

A system process that has no control region in its virtual address space and has an abbreviated context. Examples are the working set swapper and the null process. A small process is scheduled in the same manner as user processes but must remain resident until it completes execution; that is, it cannot be swapped.

SMP

See *symmetric multiprocessing*.

software-based fault tolerance

The ability to detect, isolate, and bypass faults that are executed through software.

software context

Information, residing in the control blocks, that describes the software status of a process. Examples of software context are page tables and the user's identification number. See also *software process control block*.

software interrupt

An interrupt generated on interrupt priority levels 1 through 15, which can be requested only by software.

software priority

See *process priority* and *queue priority*.

software process control block (software PCB)

The data structure used to contain a process's software context. The operating system defines a software PCB for every process when the process is created. The software PCB includes the following kinds of information about the process: current state, storage address if it is swapped out of memory, unique identification of the process, and address of the process header (which contains the hardware PCB). The software PCB resides in system region virtual address space. It is not swapped with a process.

sorting

The ordering of records in a prescribed sequence.

source file

A text file containing material suitable for translation into an object module by an assembler or compiler. Such files cannot be run or linked.

source member

Any shadow set member that is in a consistent state and can be a source for full copy or merge operations. During full copy operations, the shadowing software reads data from source members and copies the data to inconsistent members to bring them into full shadow set membership. All shadow sets must have at least one source member.

source program

A program that expresses an algorithm in a programming language such as FORTRAN, COBOL, or assembly language.

source task

The task that initiates a logical link connection request in a task-to-task communication environment.

SP

See *stack pointer*.

spanned record

A record that can cross block boundaries. A spanned record consists of one or more data segments. The position of a segment within the record and the length of the segment is denoted by the segment control word, the first five characters of each segment.

spin lock

In a VMS symmetric multiprocessing system, a semaphore associated with a set of system structures, fields, or registers whose integrity is critical to the performance of a specific operating system task. There are two types of spin lock: static and dynamic. Static spin locks are assembled permanently into the system; the same static spin locks exist in the same memory locations in all VMS multiprocessing systems. A fork lock is a form of static spin lock. Dynamic spin locks are created as required by the I/O configuration of a system; as a result, the set of dynamic spin locks differ from processor to processor. A device lock is a form of dynamic spin lock. See *fork lock* and *device lock*.

spin wait

In a VMS symmetric multiprocessing system, an execution loop performed by a processor attempting to acquire a spin lock already owned by another processor in the system. This activity is also known as a busy wait.

spool queue

The list of files, supplied by processes, that are to be processed by a symbiont. For example, a line printer queue is a list of files to be printed on the line printer.

spooling

The technique of using a high-speed mass storage device to buffer data passing between low-speed I/O devices and high-speed memory. Output spooling is the method by which output to a low-speed peripheral device (such as a line printer) is placed into queues maintained on a high-speed device (such as disk) to await transmission to the low-speed device. Input spooling is the method by which input from a low-speed peripheral (such as the card reader) is placed into queues maintained on a high-speed device (such as disk) to await transmission to a job processing that input.

SPT

See *system page table*.

SSP

See *supervisor-mode stack pointer*.

stack

An area of memory set aside for temporary storage or for procedure and interrupt service linkages. A stack uses the last-in/first-out (LIFO) concept. As items are added to ("pushed on") the stack, the stack pointer (SP) decrements. As items are retrieved from ("popped off") the stack, the SP increments.

stack frame

A standard data structure built on the stack during a procedure call, starting from the location addressed by the frame pointer (FP) to lower addresses, and popped off during a return from procedure. Also called call frame.

stack pointer (SP)

General register 14 (R14). SP contains the address of the top (lowest address) of the processor-defined stack. Reference to SP accesses one of the five possible stack pointers—kernel, executive, supervisor, user, or interrupt—depending on the value in the current mode and interrupt stack bits in the processor status longword.

standalone BACKUP

A version of the Backup Utility that is booted into memory and runs without the control of the VMS operating system.

standalone system

A computer system that is not incorporated into a network or VAXcluster system.

start I/O routine

The routine in a device driver that is responsible for obtaining necessary resources (for example, the controller data channel) and activating the device unit.

state

The functions that are currently valid for a given network component. States include line, circuit, local node, module, data terminal equipment (DTE), and logging.

state queue

A list of processes in a particular processing state. The scheduler uses state queues to keep track of processes' eligibility to execute. State queues include processes waiting for a common event flag, suspended processes, and executable processes.

static load balancing

A method of work distribution in which every process in an application is preassigned to a processor during process creation.

status

A display type for the Network Control Program (NCP) commands SHOW and LIST. Status refers to dynamic information about a component that is kept in either the volatile or permanent database.

status code

A longword value that indicates the success or failure of a specific function. For example, system services always return a status code in general register R0 upon completion.

steady state

A shadow set is in a steady state when all its members are consistent and there is no full copy operation or merge operation in progress to any members of the set.

store through

See *write through*.

stream

An access window to a file associated with a record access control block, supporting record operation requests.

stream record format

Property of a file specifying that the data in the file is interpreted as a continuous sequence of bytes, without control information. Stream record format applies to sequential files only.

string

A connected sequence of characters. When a text editor searches for a word or phrase in a text file, it is looking for a string. The character sequence that forms a command is often called a command string.

string search buffer

A storage area used to store a connected sequence of characters being searched for.

strong definition

Definition of a global symbol that is explicitly available for reference by modules linked with the module in which the definition occurs. The linker always lists a global symbol with a strong definition in the symbol portion of the map. The librarian always includes a global symbol with a strong definition in the global symbol table of a library. Contrast with *weak definition*.

strong reference

A reference to a global symbol in an object module that requests the linker to report an error if it does not find a definition for the symbol during linking. If a library contains the definition, the linker incorporates the library module defining the global symbol into the image containing the strong reference.

subdirectory

A directory file, cataloged in a higher-level directory, that lists additional files belonging to the owner of the directory.

subprocess

A subsidiary process created by another process. The process that creates a subprocess is its owner. A process and its subprocesses share a pool of quotas and limits. When an owner process is removed from the system, all its subprocesses (and their subprocesses) are also removed.

subroutine

(1) A subsidiary routine that executes when called by another program. A subroutine is often called repeatedly until a certain condition is met.

(2) A routine entered by means of a JSB or BSB instruction. Contrast with *procedure*.

substate

An intermediate circuit state that is displayed for a circuit state display when the Network Control Program (NCP) commands SHOW or LIST are entered.

summary

The default display type for the Network Control Program (NCP) commands SHOW and LIST. A summary includes the most useful information for a component, selected from the status and characteristics information.

supervisor mode

The third most privileged processor access mode (mode 2). The operating system's command interpreter runs in supervisor mode.

supervisor-mode stack pointer (SSP)

The process context stack pointer for supervisor mode.

suspension

A state in which a process is inactive but known to the system. A suspended process becomes active again when another process requests the operating system to resume it. It also becomes active to service executive mode and kernel mode ASTs. Contrast with *hibernation*.

SVA

See *system virtual address*.

swap mode

A process execution state that determines the eligibility of a process to be swapped out of the balance set. If process swap mode is disabled, the process working set is locked in the balance set.

swapper

The process that performs systemwide memory scheduling. The swapper writes modified pages to secondary storage, creates a shell for new processes, shrinks the physical size of inactive processes, removes processes from the balance set, and brings processes waiting for execution into the balance set.

swapping

The method for sharing memory resources among several processes by writing an entire working set to secondary storage (swap out) and reading another working set into memory (swap in). For example, a process's working set can be written to secondary storage while the process is waiting for I/O completion on a slow device. It is brought back into the balance set when I/O completes. Contrast with *paging*.

switch

See *qualifier*.

switched virtual circuit (SVC)

A temporary logical association between two DTEs (data terminal equipment) connected to a packet switching data network (PSDN), which is analogous to connection by a dialup line. An SVC is set up only when there is data to transmit and is cleared when the data transfer is complete.

symbiont

A process that transfers record-oriented data to or from a device. For example, an input symbiont transfers data from card readers to disks. An output symbiont transfers data from disks to line printers.

symbiont manager

The function (in the system process called the job controller) that maintains spool queues and dynamically creates symbiont processes to perform the necessary I/O operations.

symbol

An entity that when defined will represent a particular function or entity (for example, a command string, directory name, or file name) in a particular context. See *local symbol*, *global symbol*, and *universal symbol*.

symbol table

(1) The portion of an executable image that contains the definition of global symbols used by the debugger for images linked with the /DEBUG qualifier.

(2) A table in which the DIGITAL Command Language (DCL) places local symbols. DCL maintains a local symbol table for each command level.

symbolic debugger

See *debugger*.

symmetric multiprocessing (SMP)

A multiprocessing system configuration in which all processors have equal access to operating system code residing in shared memory and can perform all, or almost all, system tasks.

synchronous backplane interconnect (SBI)

The part of the hardware that interconnects the processor, memory controllers, MASSBUS adapters, and the UNIBUS adapter.

synchronous disconnect

The disconnect that occurs when a nontransparent task can issue a call to terminate I/O operations over a logical link without deassigning the channel. Thus, the task can use the channel for subsequent I/O operations with the same or a different remote task.

synchronous record operation

A mode of record processing in which a user program issues a record read or write request and then waits until that request is fulfilled before continuing to execute.

syntax

The particular form of a command, including spelling and the order of qualifiers and parameters. Misspelled words are the most common syntax errors.

SYSTEM

In the context SYSTEM/OWNER/GROUP/WORLD, SYSTEM refers to the group numbers of less than or equal to 10 (octal) that are used by operating system and its controlling users, the system operators, and the system manager.

system address space

See *system space* and *system region*.

system base register (SBR)

A processor register containing the physical address of the base of the system page table.

system communications services (SCS)

A protocol responsible for the formation and breaking of intersystem process connections and for flow control of message traffic over those connections. System services such as the VAXcluster connection manager and the mass storage control protocol (MSCP) disk server communicate with this protocol.

system control block (SCB)

The data structure in system space that contains all the interrupt and exception vectors known to the system.

system control block base register (SCBB)

A processor register containing the base address of the system control block.

system-defined identifier

One of three classes of identifiers. System-defined identifiers are provided by the system to identify groups of users according to their usage of the system. For example, all users who access the system by dialing up receive the DIALUP identifier. See also *identifier*.

system device

The random access mass storage device unit on which the volume containing the operating system software resides.

system disk

The disk that contains the VMS operating system. A VMS system disk is set up so that most of the VMS files can be shared by several VAX processors. In addition, each processor has its own directory on the system disk that contains its page, swap, and dump files.

system dynamic memory

Memory reserved for the operating system to allocate as needed for temporary storage. For example, when an image issues an I/O request, system dynamic memory is used to contain the I/O request packet. Each process has a limit on the amount of system dynamic memory that can be allocated for its use at one time.

system identification register

A processor register that contains the processor type and serial number.

system image

The image that is read into memory from disk when the system is started up.

system length register (SLR)

A processor register containing the system page table in longwords.

system manager

The person responsible for the policies, procedures, and the daily operation of a computer system. VMS system management tasks are sometimes performed by more than one person and might include responsibilities for cluster management.

system page table (SPT)

The data structure that maps the system region virtual addresses, including the addresses used to refer to the process page tables. The SPT contains one page table entry (PTE) for each page of system region virtual memory. The physical base address of the SPT is contained in the system base register.

system password

A password required by a terminal before login can be initiated at the terminal.

system queue

A queue used and maintained by operating system procedures. See also *state queue*.

system region

The third quarter of virtual address space (the S0 region); the lowest-addressed half of system space. Virtual addresses in the system region are shareable between processes. Some of the data structures mapped by system region virtual addresses are system entry vectors, the system control block (SCB), the system page table (SPT), and process page tables.

system services

A set of routines (part of the VMS operating system and used by images) to control resources, allow process communication, control I/O, and to perform other such operating system functions.

system space

The highest-addressed half of virtual address space. See also *system region*.

system virtual address (SVA)

A virtual address identifying a location in system space.

system virtual space

See *system space*.

tape mass storage control protocol (TMSCP)

The software protocol used to communicate I/O commands between a VAX processor and DSA-compliant tape devices on the system. Equivalent to mass storage control protocol (MSCP).

target node

The node that receives a memory image from another node during a downline load; a node that loops back a test message.

target task

The task that receives and processes a logical link connection request in a task-to-task communication environment.

task

(1) In networking, an image running in the context of a process.

(2) In transaction processing, a unit of work that performs a specific function and that a terminal user can select for processing. Tasks implement the business unit of work. For example, a task might book an airline reservation, update a parts inventory, or debit an account.

task specifier

Information provided to DECnet VAX software that enables it to complete a logical link connection to a remote task. This information includes the name of the remote node on which the target task runs and the name of the task itself.

terminal

The general name for peripheral devices that have keyboards and video screens or printers. Under program control, a terminal enables users to type commands and data on the keyboard and receive messages on the video screen or printer. Examples of terminals are the LA36 DECwriter hardcopy terminal, the VT100 video display terminal, and the VT240-series video terminal.

Terminal Fallback Facility (TFF)

A facility that provides table-driven character conversion for terminals.

terminal server

A communications device that connects terminals, modems, or printers to an Ethernet network.

text buffer

A text editor's storage area for text (either terminal input or file input).

TFF

See *Terminal Fallback Facility*.

thread

A single, sequential flow of control within a program. It is the active execution of a designated routine, including any nested routine invocations. A single thread has a single point of execution within it. A thread can be executed in parallel with other threads.

thread-serial service

A reentrant system service is thread serial if it blocks the current thread and all other threads that attempt to call the same service or other related services until the first call returns.

thread-synchronous service

A reentrant service is thread synchronous if it blocks only the current thread and allows other threads to execute the same operation while the current thread is blocked.

throughput

The number of transactions or jobs a computer can complete in a given period of time.

tied account

See *captive account*.

tightly coupled system

A multiprocessing system configuration consisting of multiple processors sharing a single copy of the operating system. These processors are connected so that they can communicate and share data. Contrast with *loosely coupled system*.

timeout

The expiration of the time limit in which a device is to complete an I/O transfer. The driver's wait for interrupt request specifies the timeout limit.

timer

Two system interrupt service routines: one (the hardware clock) that maintains the time of day and the date and another (the software timer) that scans for device timeouts and performs time-dependent scheduling upon request.

timesharing

A method of allocating computer time in which each process gets use of the CPU in turn. See also *real-time processing*.

timeslicing

A mechanism by which running threads are preempted at fixed intervals. Timeslicing ensures that every thread is allowed time to execute.

TMSCP

See *tape mass storage control protocol*.

TP monitor

See *Transaction Processing Monitor.*

traceback

The system facility that examines and displays the status of the user call stack when an image terminates abnormally.

track

A collection of blocks at a single radius on one recording surface of a disk.

transaction

An exchange of information between a user and a database or file. The operations in a transaction are treated as a group; either all of them are completed at once or none of them is completed. A transaction is a complete process from input to output, regardless of the number of events in between.

transaction manager

Software that coordinates the reliable completion of a transaction.

transaction processing

A technique for organizing multiple-user, high-volume, online applications that provides control over user access and updates of data.

Transaction Processing Monitor (TP monitor)

The master control executive for controlling a transaction processing system. The TP monitor includes a collection of software that schedules operations, allocates resources, and controls the operation of user and system programs which perform I/O, and updates files or databases.

A TP monitor is designed to handle medium-to-large-scale transaction processing systems that require high levels of availability, performance, and productivity.

transfer address

The address of the location containing a program entry point (the first instruction to execute).

transient state

A change of state. A shadow set is in a transient state if some of its members are undergoing either a full copy operation or a merge operation.

transition time

The amount of time (experienced as a gap in user response time) that a failed system takes to reconfigure itself following a component failure.

translation buffer

An internal processor cache containing translations for recently used virtual addresses.

translation-not-valid fault

See *page fault.*

transparent

The performance of functions not visible to the user. For example, when a command is entered, the command interpreter parses the command string and invokes the appropriate system software. The user sees only the result of the processing, not the processing itself.

transparent failover

The ability of a computing system to reconfigure itself or switch processing to a redundant component and continue processing without writing any user code or taking any corrective action.

trap

An exception condition that occurs at the end of the instruction that caused the exception. The program counter saved on the stack is the address of the next instruction that would normally have been executed. All software can enable and disable some of the trap conditions with a single instruction.

trap enables

Three bits in the processor status word (PSW) that control the processor's action on certain arithmetic exceptions.

tributary

A physical termination on a multipoint circuit that is not a control station.

tributary address

A numeric address that the control station uses to poll a tributary.

Trojan horse program

A program that gains access to otherwise secured areas through its pretext of serving one purpose when its real intent is far more devious and potentially damaging.

turnkey account

See *captive account*.

Two-Phase Commit Protocol

A protocol that DECdtm services implement to guarantee that a transaction will have a single outcome; for example, either a transaction has a permanent effect on a resource or has no effect. An example of a resource is a database system.

two's complement

A binary representation for integers in which a negative number is one greater than the bit complement of the positive number.

two-way associative cache

A cache organization that has two groups of directly mapped blocks. Each group contains several blocks for each index position in the cache. A block of data from main memory can go into any group at its proper index position. A two-way associative cache is a compromise between the extremes of fully associative and direct mapping cache organizations that takes advantage of the features of both.

type-ahead

A terminal handling technique where the user can enter commands and data while the software is processing a previously entered command. The commands typed ahead are not echoed on the terminal until the command processor is ready to process them. They are held in a type-ahead buffer.

UAF

See *user authorization file.*

UBA

See *UNIBUS adapter.*

UBI

See *UNIBUS interface.*

UCB

See *unit control block.*

UETP

See *User Environment Test Package.*

UFD

See *user file directory.*

UIC

See *user identification code.*

unblocked record

A record that is contained in a single block. No other records or parts of records are contained in that block.

UNIBUS adapter (UBA)

An interface between the backplane interconnect to the VAX–11/780 and the UNIBUS.

UNIBUS interface (UBI)

An interface between the backplane interconnect to the VAX–11/750 and the UNIBUS.

uninterrupted service

The ability of a computing system to continue providing application service during and after component failure without interruption or perceptible pause.

unit

See *device unit.*

unit control block (UCB)

A structure in the I/O database that describes the characteristics of and current activity on a device unit. The unit control block also holds the fork block for its unit's device driver; the fork block is a critical part of a driver fork process. The UCB also provides a dynamic storage area for the driver.

unit initialization routine

The routine that readies controllers and device units for operation. Controllers and device units require initialization after a power failure and during the driver loading procedure.

unit record device

A device such as a card reader or line printer.

universal symbol

A global symbol in a shareable image that can be used by modules linked with that shareable image. Universal symbols are typically a subset of all the global symbols in a shareable image. When creating a shareable image, the linker ensures that universal symbols remain available for reference after symbols have been resolved.

unwind the call stack

To remove call frames from the stack by tracing back through nested procedure calls using the current contents of the frame pointer (FP) register and the FP register contents stored on the stack for each call frame.

upline dump

A DECnet VAX function that allows an adjacent node to dump its memory to a file on a VMS system.

urgent interrupt

An interrupt received on interrupt priority levels 24 through 31. These can be generated only by the processor for the interval clock (in certain VAX systems), serious errors, and power failure.

user authorization file (UAF)

A file containing an entry for every user that the system manager authorizes to gain access to the system. Each entry identifies the user name, password, default account, user identification code (UIC), quotas, limits, and privileges assigned to individuals who use the system.

User Environment Test Package (UETP)

A collection of routines that verify that the hardware and software systems are complete, properly installed, and ready to use.

user file directory (UFD)

A file that briefly catalogs a set of files stored on disk or tape. The directory includes the name, type, and version number of each file in the set. It also contains a unique number that identifies that file's actual location and points to a list of its file attributes. See also *directory*.

user identification code (UIC)

A 32-bit value assigned to users and to files, global sections, common event flag clusters, and mailboxes that specifies the type of access (read and/or write access and, in the case of files, execute and/or delete access) available to the SYSTEM, OWNER, GROUP, and WORLD. A UIC has two formats: numeric and alphanumeric. The numeric UIC consists of a group identifier and a member identifier separated by a comma and enclosed within square brackets. These

identifiers may appear as alphanumeric characters. The alphanumeric UIC consists of a member name and, optionally, a group name.

user mode

The least privileged processor access mode (mode 3). User processes and Run-Time Library (RTL) procedures run in user mode.

user-mode stack pointer (USP)

The process context stack pointer for user mode.

user name

The name that a user types on a terminal to log in to the system. See also *password*.

user number

See *member number*.

user password

See *password*.

user privileges

The privileges granted a user by the system manager. See *process privileges*.

USP

See *user-mode stack pointer*.

utility

A program that provides a set of related general-purpose functions, such as a program development utility (an editor, a linker), a file management utility (file copy or file format translation program), or operations management utility (disk quotas, diagnostic program).

value return registers

The general registers R0 and R1 used by convention to return function values. These registers are not preserved by any called procedures. They are available as temporary registers to any called procedure. All other registers (R2 through R11, AP, FP, SP, PC) may be preserved across procedure calls.

variable-length bit field (VBF)

A set of 0 to 32 contiguous bits located arbitrarily with respect to byte boundaries. A variable bit field is specified by four attributes: the address A of a byte; the bit position P of the starting location of the bit field with respect to bit 0 of the byte at address A; the size, in bits, of the bit field; and an indication whether the field is signed or unsigned.

variable-length record format

A file format in which records may be of different lengths.

variable-length with fixed-length control field (VFC) record format

A file format in which records of variable length contain an additional fixed-length control area. The control area may be used to contain file line numbers and print format controls.

VAXBI

The part of the VAX 8200/8250/8300/8350 hardware that connects I/O adapters with memory controllers and the processor. In a VAX 8530/8550/8700/8800 or VAX 6200/6300 system, the part of the hardware that connects I/O adapters with the bus that interfaces with the processor and memory.

VAX

Virtual Address Extension.

VAX Vector Instruction Emulation Facility (VVIEF)

A standard feature of the VMS operating system that allows vectorized applications to be written and debugged in a VAX system in which vector processors are not available. VVIEF emulates the VAX vector processing environment, including the nonprivileged VAX vector instructions and the VMS vector system services. Use of VVIEF is restricted to user mode code.

VAXcluster configuration

A highly integrated organization of VMS systems that communicate over a high-speed communications path. VAXcluster configurations have all the functions of single node systems, plus the ability to share CPU resources, queues, and disk storage. Like a single-node system, the VAXcluster configuration provides a single security and management environment. Member nodes can share the same operating environment or serve specialized needs.

VAXclusters Phase II

See *Phase II VAXcluster configuration.*

VAXcluster system

A loosely coupled, highly integrated, distributed computing environment. There are three types of VAXcluster system configurations, depending on the medium used for interprocessor communications: CI-based, local area, and mixed-interconnect.

VBF

See *variable-length bit field.*

VBN

See *virtual block number.*

VCB

See *volume control block.*

vector

(1) A storage location (known as an interrupt or exception vector) that contains the starting address of a procedure to be executed when a given interrupt or exception occurs. The system defines separate vectors for each interrupting device controller and for classes of exceptions. Each system vector is a longword.

(2) For the purposes of exception handling, users can declare up to two software exception vectors (primary and secondary) for each of the four access modes. Each vector contains the address of a condition handler.

(3) A one-dimensional array.

(4) A group of related scalar values, or elements, all of the same data type.

vector capability

A software abstraction by which the VMS operating system makes system vector processing resources available to a process. A system manager can restrict the use of the vector processor to users holding a particular identifier by associating an access control list (ACL) with the CAPABILITY object VECTOR.

vector-capable system

A VAX system that, in its hardware implementation, complies with the VAX vector architecture. A vector-capable system that incorporates one or more optional vector processors is known as a *vector-present system*.

vector consumer

A process executing an image that issues VAX vector instructions and, thus, requires the VECTOR capability and has a vector context. VMS *must* schedule a vector consumer on a scalar-vector processor pair in the VAX system. As long as it remains a vector consumer, a process is effectively prohibited from executing on any scalar processor in the system. See *scalar consumer* and *marginal vector consumer*.

vector context

Current vector state of a process. See *vector state*.

vector-present system

A VAX system that, in its hardware implementation, complies with the VAX vector architecture, and incorporates one or more optional vector processors. See *vector-capable system*.

vector processor

Specialized computer hardware that executes vector instructions. For example, hardware components that implement the VAX vector architecture.

vector state

The contents of the vector registers V0 through V15, the contents of the vector control registers, the vector processor status, and the vector exception state, as associated with a given thread of execution. A process may have several vector states: for instance, one associated with the mainline thread of execution and another associated with an AST service routine. The current vector state is referred to as the vector context of a process. See *vector context*.

vectorized program

An application that makes use of a vector processor. There are several methods by which an application may be vectorized. For instance, one of its modules may have been compiled by a vectorizing compiler or call one or more vectorized routines. VAX MACRO programs that directly issue VAX vector instructions are also considered vectorized programs.

version number

(1) The field following the file type in a file specification. It begins with a semicolon or period and is followed by a number which generally identifies it as the latest file created of all files having the identical file specification.

(2) The number used to identify the revision level of program.

VFC

See *variable-length with fixed-length control field record format.*

video terminal

A terminal with a video screen for accepting output. See *terminal.*

virtual address

A 32-bit integer identifying a byte location in the process, control, and system regions of virtual memory. The memory management hardware translates a virtual address to a physical address. The term virtual block number (VBN) refers to the address used to identify a virtual block on a mass storage device. See also *virtual address space.*

virtual address space

The set of all possible virtual addresses that an image executing in the context of a process can use to identify the location of an instruction or data. The VMS virtual address space seen by the programmer is a linear array of 4,294,967,296 (2^{32}) byte addresses. The VMS system distinguishes between the physical memory required by a process and the virtual address space that the process defines. A process's virtual address space is the range of memory locations that the process can address.

virtual block number (VBN)

The file-relative address of a block on a mass storage device. The first block in a file is always virtual block 1. Contrast with *logical block number* and *physical block number.*

virtual circuit

An association between two DTEs (data terminal equipment) connected to a packet switching data network (PSDN), whereby they are able to interact as if a specific circuit were dedicated to them throughout the transmission. In reality a logical connection is established, the actual physical circuits being allocated according to route availability, overload conditions, and so on.

virtual disk

A dedicated portion of disk space allocated for storing console files when performing save or restore operations on console media. Virtual disks eliminate the need to create a disk or directory for the console media. Also, because the command procedure that handles the task invokes the Exchange Utility, the conversion of the file format from RT–11 to Files–11 is done automatically.

virtual I/O functions

A set of I/O functions that must be interpreted by an ancillary control process.

virtual memory

The set of storage locations in physical memory and on disk that is referred to by virtual addresses. From the programmer's viewpoint, the secondary storage locations appear to be locations in physical memory. The size of virtual memory in any system depends on the amount of physical memory available and the amount of disk storage used for nonresident virtual memory.

Virtual Memory Boot (VMB)

The primary bootstrap program that initializes a VAX system with VMS. The program is located in the boot block of physical memory and is the first program read by the CPU when a processor is booted after it has been halted.

virtual page number (VPN)

The virtual address of a page of virtual memory.

virtual unit

The shadowing software-created representation of the devices that compose a shadow set. The virtual unit receives and directs I/O requests to the shadow set members.

VMB

See *Virtual Memory Boot*.

VMS

Virtual Memory System.

VMS Lock Manager

A VMS facility that mediates lock requests.

volatile database

A memory image that contains information about network management components.

volume

A mass storage media such as a disk pack or reel of magnetic tape. The volume is the largest logical unit of the file structure.

volume control block (VCB)

A data structure that contains the information needed to control access to a volume. It is created when the volume is mounted.

volume set

The file-structured collection of data residing on one or more mass storage media.

volume shadowing

The process of maintaining multiple copies of the same data on two or more disk volumes. When data is recorded on more than one disk volume, you have access to critical data even when one volume is unavailable.

VPN

See *virtual page number*.

VVIEF

See *VAX Vector Instruction Emulation Facility*.

waiting

Becoming inactive. A process enters a process wait state when the process suspends itself, hibernates, or declares that it needs to wait for an event, resource, mutex, and so forth.

wait for interrupt request

A request made by a driver's start I/O routine after it activates a device. The request causes the driver fork process to be suspended until the device requests an interrupt or the device times out.

waking

Activating a hibernating process. A hibernating process can be awakened by a time-scheduled wake-up call.

watchpoint

A watchpoint is a memory address, register, or (typically) a variable declared in a program whose value is monitored during program execution.

WCB

See *window control block*.

WCS

See *writable control store*.

WDCS

See *writable diagnostic control store*.

weak definition

Definition of a global symbol that is not explicitly available for reference by modules linked with the module in which the definition occurs. The librarian does not include a global symbol with a weak definition in the global symbol table of a library. Weak definitions are often used when creating libraries to identify those global symbols that are needed only if the module containing them is otherwise linked with a program. Contrast with *strong definition*.

weak reference

A reference to a global symbol that requests the linker not to report an error or search the default library's global symbol table to resolve the reference if the definition is not in the modules explicitly supplied to the linker. Weak references are often used when creating object modules to identify those global symbols that may not be needed at run time.

wildcard character

A nonalphanumeric character such as an asterisk or percent sign that is used within, or in place of, a file name, file type, directory name, or version number in a file specification to indicate "all" for the given field.

window

(1) See *mapping window*.

(2) A range of packets authorized for transmission across an X.25 data terminal equipment/data circuit-terminating equipment (DTE/DCE) interface. The lowest sequence number in the window is referred to as the lower window edge (0 when the virtual circuit is just established). The packet send sequence number of the first data packet not authorized to cross the interface is the value of the upper window edge (that is, the lower window edge plus the window size).

window control block

A data structure that stores access control information for a file. It is created when a file is accessed on a volume. It is deleted when the file is closed.

word

Two contiguous bytes (16 bits) starting on an addressable byte boundary. Bits are numbered from the right, 0 through 15. A word is identified by the address of the byte containing bit 0. When interpreted arithmetically, a word is a two's complement integer with significance increasing from bit 0 to bit 14. If interpreted as a signed integer, bit 15 is the sign bit. The value of the integer is in the range –32,768 to 32,767. When interpreted as an unsigned integer, significance increases from bit 0 through bit 15 and the value of the unsigned integer is in the range 0 through 65,535.

working set

The set of pages in process space to which an executing process can refer without incurring a page fault. The working set must be resident in memory for the process to execute. The remaining pages of that process, if any, are either in memory and not in the process working set or they are on secondary storage.

working set swapper

See *swapper*.

WORLD

In the context SYSTEM/OWNER/GROUP/WORLD, WORLD refers to all users, including the system operators, the system manager, and users both in an owner's group and in any other group.

writable control store

A hardware component of certain VAX processors that allows a customer to customize selected machine functions.

writable diagnostic control store

A hardware component of certain VAX processors that contains basic instruction microcode and diagnostic microcode.

writing

The act or capability of an image to send data. For example, when a PRINT command is entered, the specified file is read from wherever it is stored and then written to the line printer.

write access type

The specified operand of an instruction or procedure is only written during that instruction's or procedure's execution.

write allocate

A cache management technique in which cache is allocated on a write miss as well as on the usual read miss.

write back

A cache management technique in which data from a write operation to cache is copied into main memory only when the data in cache must be overwritten. This results in temporary inconsistencies between cache and main memory. Contrast with *write through*.

write miss

An event in which a write operation cannot be serviced by the processor's hardware cache.

write through

A cache management technique in which data from a write operation is copied in both cache and main memory. Cache and main memory data are always consistent. Contrast with *write back*.

X.3

A CCITT recommendation that specifies the packet assembly/disassembly (PAD) facility in a public data network.

X.25

A CCITT recommendation that specifies the interface between DTEs (data terminal equipment) and DCEs (data circuit-terminating equipment) for equipment operating in the packet mode on public data networks.

X.28

A CCITT recommendation that specifies the data terminal equipment/data circuit-terminating equipment (DTE/DCE) interface for a start-stop mode DTE accessing the packet assembly/disassembly (PAD) facility in a public data network situated in the same country.

X.29

A CCITT recommendation that specifies procedures for the exchange of control information and user data between a packet-mode DTE (data terminal equipment) and a packet assembly/disassembly (PAD) facility.

X.29 terminal

A terminal connected to a packet assembly/disassembly (PAD) facility.

XAB

See *extended attribute block*.

XQP

See *extended QIO processor*.

Y-connector

Hardware that joins two synchronous communications lines into a single output line.

zone

A section of a fully configured VAXft fault-tolerant computing system that contains a minimum of a CPU module, memory module, I/O module, and associated devices. A VAXft system consists of two such zones with synchronized processor operations. If one zone fails, processing continues uninterrupted through automatic failover to the other zone.

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-DEC-DEMO (800-332-3366) using a 1200- or 2400-baud modem. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local Digital subsidiary or approved distributor
Internal[1]	————	USASSB Order Processing - WMO/E15 *or* U.S. Area Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

[1]For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

VMS Glossary

AA–LA03B–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page Description

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

Do Not Tear - Fold Here and Tape

digital™

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Information Products
ZK01–3/J35
110 SPIT BROOK RD
NASHUA, NH 03062–9987

Do Not Tear - Fold Here

Reader's Comments

VMS Glossary

AA–LA03B–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page Description

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone

Do Not Tear - Fold Here and Tape

digital™

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Information Products
ZK01–3/J35
110 SPIT BROOK RD
NASHUA, NH 03062–9987

Do Not Tear - Fold Here

Guide to VMS Files and Devices

Order Number: AA–LA06A–TE

April 1988

This guide describes procedures for using files and devices with the VMS operating system. The tasks described in this manual are oriented toward the use of private media.

Revision/Update Information: This revised guide supersedes the *Guide to VAX/VMS Disk and Magnetic Tape Operations*, Version 4.4.

Software Version: Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital™

ZK3389

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.
In New Hampshire, Alaska, and Hawaii call 603-884-6660.
In Canada call 800-267-6215.
*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).
Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by DIGITAL. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use DIGITAL-supported devices, such as the LN03 laser printer and PostScript™ printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

™ PostScript is a trademark of Adobe Systems, Inc.

Contents

INDEX

FIGURES

TABLES

Preface

The *Guide to VMS Files and Devices* describes some of the routine tasks that general users perform on files and devices.

Intended Audience

This guide is intended for all general users.

The *Guide to VMS Files and Devices* is not a guide for system managers; it is designed primarily for the private user. System managers should refer to *Introduction to VMS System Management* and *Guide to Setting Up a VMS System*.

If you are a novice user, you can use this guide as a stepping stone to a basic understanding of file operations on disk and magnetic tape devices. Experienced users unfamiliar with DIGITAL software can use it to gain familiarity with DIGITAL terms and techniques. Experienced DIGITAL users should find the guide useful because it describes in detail most of the disk and magnetic tape operations routinely performed on the VMS operating system.

Document Structure

The chapters in this guide contain the following information:

- Chapter 1 describes basic file and magnetic tape device concepts.
- Chapter 2 describes the protection schemes that apply to file and device media.
- Chapters 3-5 describe tasks frequently performed on private disk and tape media. These chapters include examples of command procedures designed to simplify the use of routine file and device operations.
- Appendix A describes the Files–11 On-Disk Structure (ODS–2) for VMS disk files and volumes.
- Appendix B describes formats for VMS magnetic tape volumes and files.

Associated Documents

The tasks described in this guide employ various VMS utilities and DIGITAL Command Language (DCL) commands. Detailed descriptions of the utilities and commands are provided in the VMS utilities volumes and in the *VMS DCL Dictionary*.

Your specific operations guide provides instructions for procedures used in system installation.

The *Guide to Setting Up a VMS System* provides task-oriented instructions for maintaining public files and volumes; these instructions include steps for bootstrapping and running standalone BACKUP.

The *VMS Local Area VAXcluster Manual* provides information for maintaining public files and volumes in a VAXcluster environment.

The *Guide to VMS System Security* describes security features available through the VMS operating system; see this guide for more information on data protection.

The *VAX Volume Shadowing Manual* (not part of the VMS document set) describes how to mount, dismount, and maintain volumes using the volume shadowing option.

Conventions

Convention	Meaning
RET	In examples, a key name (usually abbreviated) shown within a box indicates that you press a key on the keyboard; in text, a key name is not enclosed in a box. In this example, the key is the RETURN key. (Note that the RETURN key is not usually shown in syntax statements or in all examples; however, assume that you must press the RETURN key after entering a command or responding to a prompt.)
CTRL/C	A key combination, shown in uppercase with a slash separating two key names, indicates that you hold down the first key while you press the second key. For example, the key combination CTRL/C indicates that you hold down the key labeled CTRL while you press the key labeled C. In examples, a key combination is enclosed in a box.
$ SHOW TIME 05-JUN-1988 11:55:22	In examples, system output (what the system displays) is shown in black. User input (what you enter) is shown in red.
$ TYPE MYFILE.DAT . . .	In examples, a vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to a command is shown or that not all the data a user would enter is shown.
input-file, . . .	In examples, a horizontal ellipsis indicates that additional parameters, values, or other information can be entered, that preceding items can be repeated one or more times, or that optional arguments in a statement have been omitted.
[logical-name]	Brackets indicate that the enclosed item is optional. (Brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

Summary of New and Changed Features

The *Guide to VMS Files and Devices* contains much of the same information as the Version 4.4 *Guide to VAX/VMS Disk and Magnetic Tape Operations.* The title and content have been modified to shift the focus to a more general user-oriented style.

Enhanced DCL Commands

The following DCL commands have been enhanced for VMS Version 5.0:

- ANALYZE/RMS_FILE/STATISTICS displays RMS file characteristics information for determining:
  - Whether journaling is enabled
  - The global buffer count
  - Whether file monitoring is enabled
- DIRECTORY/FULL displays whether journaling is enabled.
- SHOW DEVICES displays changes to system output.
- SHOW MAGTAPE displays change to odd parity line format.

1 Introduction to Files and Devices

This chapter provides an introduction to storage media and describes basic file and device concepts. It also discusses the importance of identifying routine operations performed on files and devices and the advantages of using command procedures to execute them.

1.1 The History of File Storage Media

As technology has progressed, so has the volume of information that must be saved. Business and industrial concerns, for instance, have compiled many types of data files about a wide range of subjects. For many years, all these data files were stored on paper in desk drawers and filing cabinets. But, as these paper files grew, it often took longer to locate the needed data than to create it in the first place. Storing all of a company's data files on paper became impractical.

As the need for saving data increased, the need also arose for a better medium for storing and retrieving data quickly, reliably, and economically. The computer and computerized filing systems provide such storage media.

At first, computerized files consisted of collections of punched cards, which provide a means of grouping related pieces of information. This information might represent, for example, a business event such as a purchase or sale of office furniture. Or, in the engineering environment, the information could represent variable equations and data constants related to stress analysis. This information, grouped on a single card, represents a record of that event. Records of similar events, grouped together, constitute a file.

As a storage medium, cards have certain advantages. They are easy to add, delete, or rearrange. However, cards become worn, require physical handling, and are bulky. Cards are also relatively slow to process because they allow only sequential access. Sequential access means that the search for a record starts at the beginning of the file and proceeds in order through each record. At times, when the needed record (or group of records) is near the end of the file, the search wastes computer processing time.

The introduction of magnetic tape as a storage medium eliminated some of the disadvantages of cards. Magnetic tape provides both a storage medium and a means for input/output (I/O), and its uses have grown along with the needs of the user. Magnetic tape offers virtually unlimited storage. It requires much less storage space than a card file with a comparable amount of data. However, magnetic tape is limited as a storage medium because, like a card file, it allows only sequential access.

In contrast to magnetic tape, disk storage allows direct file access. Direct access means that the computer can locate the desired record without first searching the records that precede it in the file. The time needed to access the record is independent of the record's location in the file.

1.2 Basic Device Concepts

This section describes basic disk and magnetic tape device concepts for VMS systems.

1.2.1 Disk Concepts

VMS files reside on Files–11 On–Disk Structure volumes. The term *Files–11 On–Disk Structure* refers to the logical structure given to the disk; namely, a hierarchical organization of files, their data, and the directories needed to gain access to them. The VMS file system implements the disk structure and provides access control to the files located on the disk. This section describes the Files–11 On–Disk Structure levels and defines the terminology related to it. (The term *Files–11* used alone always refers to *File–11 On–Disk Structure levels*.)

The smallest addressable unit of information on a disk is a block. Files–11 On-Disk Structures define a block to consist of 512 8-bit bytes. Blocks can be treated as units for transfer between a Files–11 disk volume and memory.

Blocks are logically grouped into clusters, which are the basic unit by which disk space is allocated. The system manager or operator determines the number of blocks in a cluster when a given disk, known as a volume, is first prepared for use (initialized). Cluster sizes from 1 to 65,535 blocks are allowed, but the smaller cluster sizes in the range are more practical. In general, a disk with a relatively small number of blocks is given a smaller cluster size, while larger disks are given larger cluster sizes to minimize the overhead for disk space allocation.

Contiguous clusters allocated to a particular file have been given the name *extent*. An extent can contain all or part of a file. If enough contiguous area is available on the disk, the entire file is allocated as a single extent. Sometimes, however, not enough contiguous area is available to contain the entire file, or, when you create a file initially, you may not wish to reserve the entire required amount of space. When the file is eventually extended, it is unlikely that the adjacent clusters will still be unallocated. If the adjacent clusters are already allocated to another file, the extension will not occur contiguously. Whether the clusters are contiguous or not, the file is divided into two or more parts, and each part is an extent. Thus, a file can consist of multiple extents located in separate areas on the disk, as shown in Figure 1–1. Note that the file extensions are done automatically.

Figure 1–1 File Extents

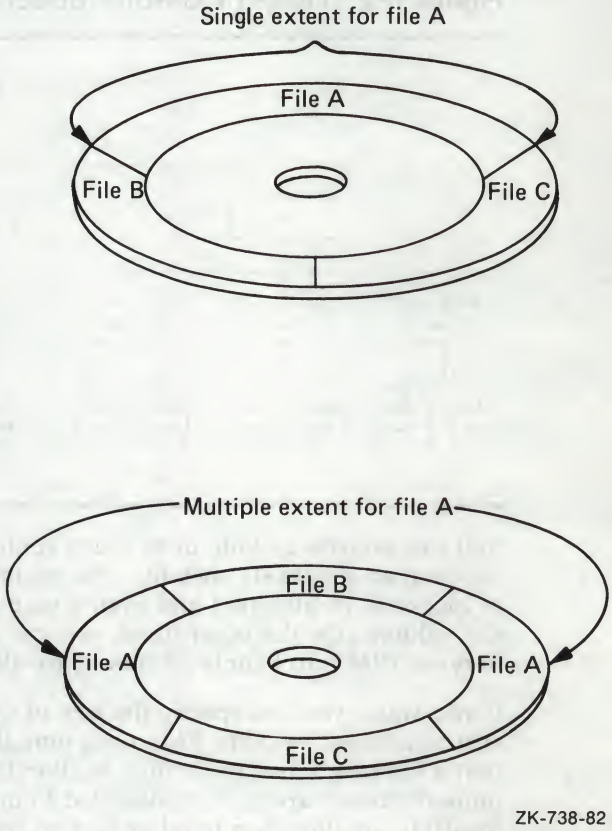
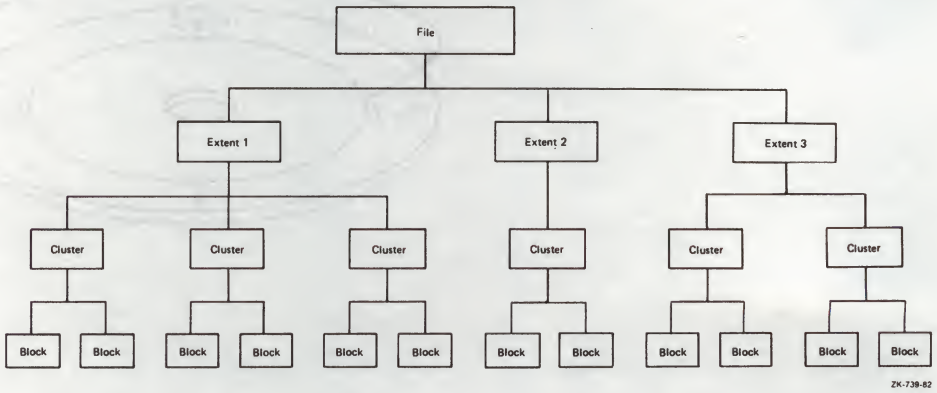


Figure 1–2 shows the hierarchy of blocks, clusters, extents, and files in the Files–11 On-Disk Structure.

Figure 1–2 Files–11 On-Disk Structure Hierarchy



You can exercise as little or as much control as you want over the allocation of space on a Files–11 disk file. For example, you can specify the number of blocks to be allocated and even give the exact location for the blocks on the volume. On the other hand, you can allow VMS Record Management Services (RMS) to handle all disk space allocation details automatically.

If you want, you can specify the size of the initial space allocation and the size to be used by VMS RMS each time the file is extended. If you find that a file needs less space than is allocated to it, you can specify that the unused clusters are to be deallocated from the file. These clusters will then be available for allocation to other files on the volume.

When a large amount of file storage space is needed, you can combine several Files–11 volumes into what is called a volume set. A volume set, although composed of several physical volumes, has the appearance of one large volume. The different extents of a file can be located on different volumes in the volume set. In general, you need not specify a particular volume in the set to locate a file or create a new one, although it is possible (and sometimes desirable for performance reasons) to specify a particular volume to be used for a certain allocation request.

The smallest unit discernible to the Files–11 structure is the sector; for most Files–11 disks, a sector is equivalent to a block, which is 512 bytes. Other basic terms related to disks are track and cylinder. A track is the collection of sectors (or blocks on Files–11 structures) at a single radius on one recording surface of a disk. It is accessible to a given read/write head position on the disk device. A cylinder consists of all tracks at the same radius on all recording surfaces of a disk.

Because access to any of the blocks in a given cylinder does not require any movement of the disk's read/write heads, it is generally advantageous to keep related data blocks in the same cylinder. For this reason, when choosing a cluster size for a large-capacity disk, a system manager often selects a cluster size that divides evenly into the cylinder size.

Figure 1–3 Tracks and Cylinders

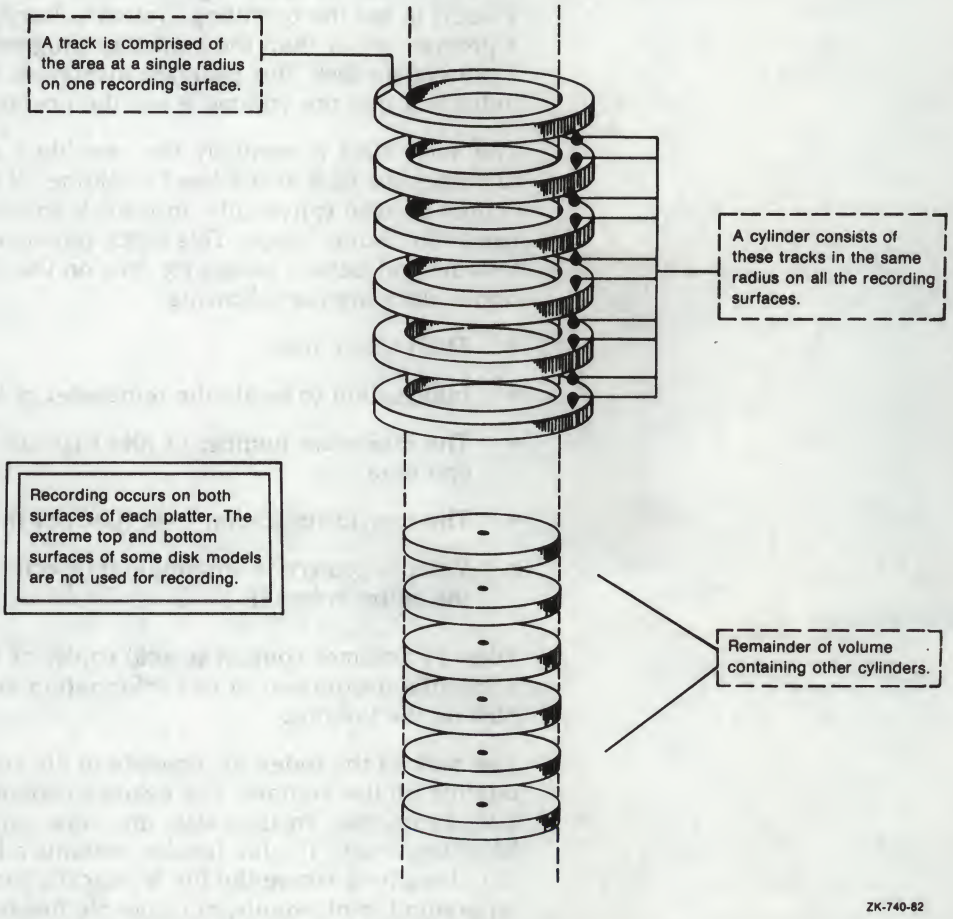


Figure 1–3 is a graphic representation of tracks and cylinders.

The remainder of this section contains a brief explanation of some basic elements of the Files–11 structure.

A Files–11 structure resides on a volume, which is a physical medium such as a disk pack. A Files–11 volume is an ordered set of 512-byte blocks. The blocks are numbered consecutively from 0 to *n—1;* the value of *n—1* is the size of the disk in blocks.

Each Files–11 volume has an index file, which is created when the volume is initialized. (You cannot use a disk as a Files–11 disk until it has been initialized with the INITIALIZE command.) The index file contains the following information:

- Bootstrap block
- Home block
- File headers

The *bootstrap block* contains the bootstrap program and is physically the first block on the volume. All Files–11 volumes have an area for this bootstrap block even if the operating system does not require a bootstrap block. If the volume is not the operating system volume, the bootstrap block area contains a program other than the bootstrap program. If an attempt is made to boot a nonsystem disk, this program displays a message on the system console indicating that the volume is not the operating system volume.

The *home block* is normally the next block after the bootstrap block; it identifies the disk as a Files–11 volume. If for some reason the home block cannot be read (physically unusable), an alternative block will be selected for use as the home block. This block provides specific information about the volume and default values for files on the volume. Among the items in the home block are the following:

- The volume name
- Information to locate the remainder of the index file
- The maximum number of files that can be present on the volume at any one time
- The user identification code (UIC) of the owner of the volume
- Volume protection information (specifies which users can read or write the entire volume)

Files–11 volumes contain several copies of the home block to ensure against accidental destruction of this information and the consequent loss of access to files on the volume.

The bulk of the index file consists of *file headers*; each file header describes one file on the volume. File headers contain information such as the owner UIC, protection, creation date and time, and Access Control Lists (ACLs). Most important, the file header contains a list of extents that make up the file, describing where the file is logically located on the volume. If a file has a large number of extents, multiple file headers may be used to describe them. A file identifier number is associated with each file header.

When you create a file, you normally specify a file name to VMS RMS, which assigns this name to the file on a Files–11 volume. RMS places the file name and file identifier associated with the newly created file in a directory, which contains an entry defining the location for each file. When you access the file, you supply the file name, which supplies a path to the file identifier through the directory entry. The file identifier, in turn, points to the location of the file header, which contains a listing of the extent or extents that locate the actual data.

1.2.2 Magnetic Tape Concepts

The VMS file storage system for magnetic tapes is based on the standard magnetic tape structure as defined by the American National Standard X3.27-1978 (referred to as the ANSI standard throughout this manual) and also supports the ISO 1001-1979 standard.

Magnetic tape data is organized sequentially; the data records and files are organized in the order in which they are written.

On VMS systems, characters of data on magnetic tape are most commonly measured in bits per inch (bpi). This measurement is called density. (The ANSI standard uses characters per inch (CPI), which is equivalent to the bpi convention used by VMS.) A 1600-bpi tape can accommodate 1600 characters of data, using an inch of tape.

Even though a magnetic tape may have a density of 1600 bpi, there are not always 1600 characters on every inch of tape because of the interrecord gap (IRG). The IRG is an interval of blank space between data records; it is created automatically when records are written to the magnetic tape. The IRG is a breakpoint on the magnetic tape, which allows the magnetic tape unit to decelerate and stop, if necessary, after a record operation and then accelerate to working speed for the next record operation.

Each IRG is approximately 0.6 inch in length (this length varies with the type of tape drive). Writing an 80-character record at 1600 bpi requires 0.05 inch of space. Therefore, the IRG requires 12 times more space than the data, wasting valuable storage space. RMS can reduce the size of this wasted space by using record blocking. This technique groups individual records into a block and places the IRG after the block rather than after each record.

Note that a block on magnetic tape is different from a block on disk. On disk, a block is fixed at a size of 512 bytes; on magnetic tape, the size of a block is determined by the user. However, record blocking requires more buffer space to be allocated for your program, which increases RMS memory requirements. The greater the number of records in a block, the greater the buffer size requirements. You must determine the point at which the benefits of record blocking cease. This determination should be based on the configuration of your computer system and your authorized environment.

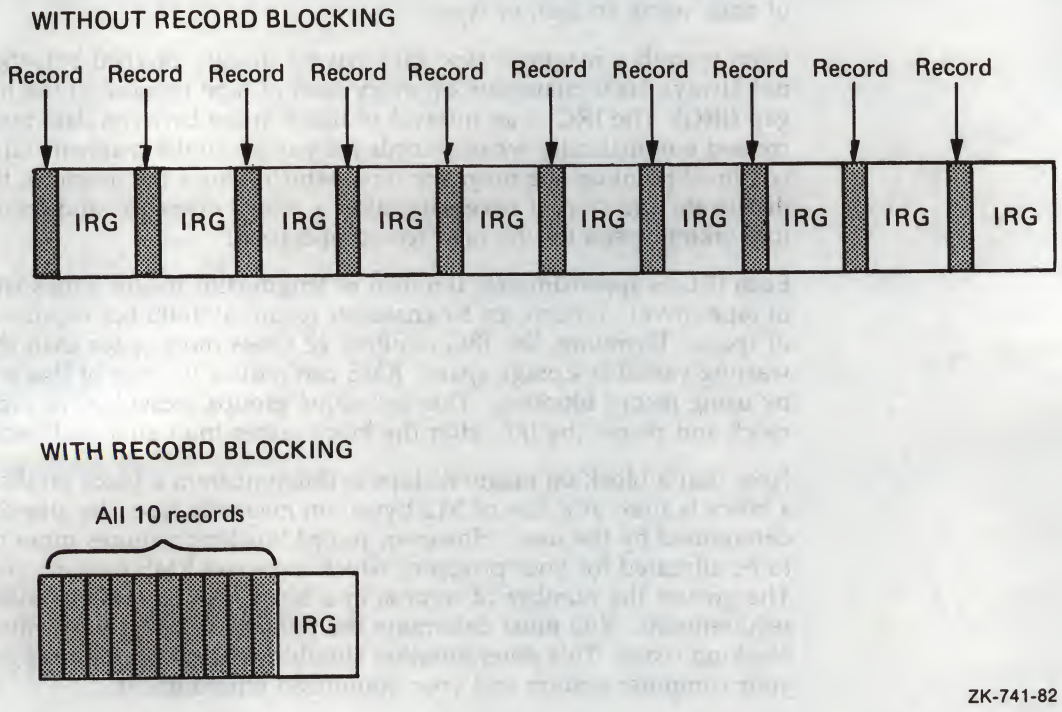
Figure 1–4 shows how space can be saved by record blocking. Assume that a 1600-bpi magnetic tape contains 10 records that are not grouped into a block. Each record is 160 characters long (0.1 inch at 1600 bpi) with a 0.6 inch IRG after each record. This uses 7 inches of tape. However, placing the same 10 records into one block uses only 1.6 inches of tape (1 inch for the data records and 0.6-inch for the IRG).

Record blocking also increases the efficiency of the flow of data into the computer. For example, ten unblocked records require ten separate physical transfers, while ten records placed into a single block require only one physical transfer. Moreover, a shorter length of magnetic tape is traversed for the same amount of data; thus, the operation is completed in less time.

Data on magnetic tape is also organized into files. When you create a file on magnetic tape, the magnetic tape file system writes a set of header labels on the tape immediately preceding the data blocks. These labels contain information such as the user-supplied file name, creation date, and expiration date. Additional labels, called trailer labels, are also written following the file. To access a file on magnetic tape by the file name, the file system searches the tape for the header label set that contains the specified file name.

When the data blocks of a file or related files do not physically fit on one volume (a reel of magnetic tape), the file is continued on another volume. This is a multivolume file. The volumes in a multivolume file constitute a volume set. When access is made to a file on a volume set, all volumes in the set are accessible.

Figure 1–4 Interrecord Gaps



1.3 Using Command Procedures to Perform Routine File and Device Operations

Many of the operations that you perform on disk and magnetic tape media are routine in nature. Therefore, you will find it worthwhile to take the time to identify those tasks that you routinely perform at your particular site. Once you have isolated those tasks, you can design command procedures to assist you in performing them.

If you are a system manager or an operator, for example, you must frequently perform data integrity tasks such as backing up media. You could enter all of the commands, parameters, and qualifiers required to back up your media each time that you perform the backup operation, or you can write a single command procedure (containing that set of commands, qualifiers, and parameters) that, when executed, would also perform the backup operation.

In order to familiarize yourself with the syntax used to design and execute command procedures, see the *Guide to Using VMS Command Procedures.*

2 File and Device Protection

The VMS operating system provides various types of protection. This chapter focuses on data, file, and device protection for VMS systems.

2.1 Data Protection

The VMS operating system protects data on disk and magnetic tape volumes to ensure against accidental or unauthorized access. The term *volume* refers to the entity that exists when a medium is mounted on a device. For example, disk packs and reels of tape are called volumes when they are mounted on disk and magnetic tape drives.

VMS systems support the protection of data on disk and magnetic tape media at the volume and file levels. At the volume level, the system provides protection for both disks and tapes. At the file level, the system provides protection for individual disk files, directory files that reside on disk volumes, and tape files. However, distinct file-level protection for magnetic tape files is limited to one special case, described in Section 2.2.2.4.

In addition to protecting the data on mounted volumes, the system also provides device-level protection. For more information on setting device protection characteristics, see the descriptions of the DCL commands INITIALIZE, MOUNT, and SET VOLUME in the *VMS DCL Dictionary*.

On VMS, data residing on disk and tape volumes can be protected by one or more of the following:

- User identification codes (UICs)
- Access control lists (ACLs)
- ANSI-standard accessibility (magnetic tape only)

2.1.1 User Identification Code (UIC)-Based Protection

User Identification Code (UIC)-based protection is supported for disk and magnetic tape volumes and for individual files on disk volumes, including directories. UIC-based protection is determined by an owner's user identification code and a protection code. The owner UIC is normally the UIC of the user who created the file or volume. The protection code indicates who is allowed access and for what purposes.

UIC formats can contain alphanumeric characters or they can consist entirely of octal numbers. For example, [VMS,USER] and [360,030] are both legitimate UIC formats.

When a user attempts to access a file or volume, the user's UIC is compared against the owner UIC of the file or volume. Depending on the relationship of the UICs, the user falls into one or more of the following categories:

- SYSTEM — All users who have system privilege (SYSPRV) or low group numbers, usually from 1 through 10 (octal). However, the exact range of system group numbers is determined by the system manager when

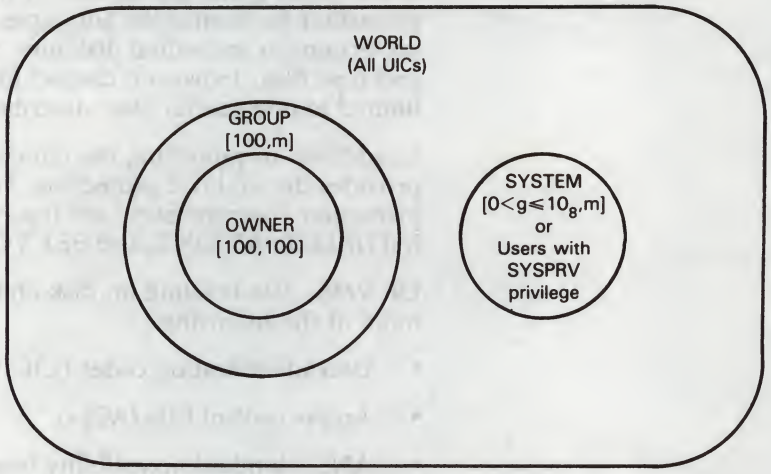
the system is generated and may range as high as 37776 (octal). These group numbers are generally for system managers, system programmers, and operators. In addition, the SYSTEM protection field is used for those users with the user privilege GRPPRV, whose UIC group matches the group of the file or volume owner UIC.

- OWNER — The user with the same UIC as the person who created, and therefore owns, the volume or file
- GROUP — All users, including the owner, who have the same group number in their UICs as the owner of the file
- WORLD — All users including those in the first three categories

Note: SYSTEM and OWNER categories always have access to magnetic tape volumes, except as noted in Section 2.1.3.

Figure 2–1 illustrates the relationships of these categories to each other.

Figure 2–1 Illustrating User Categories with a UIC of [100,100]



NOTE: THE SYSTEM MANAGER CAN EXTEND THE SYSTEM GROUP NUMBER LIMIT TO 37776$_8$

ZK-778-82

Each user category is allowed or denied five types of access:

- READ
- WRITE
- EXECUTE
- DELETE
- CONTROL

The meaning of each access type depends on the object to which it is applied (disk or magnetic tape volume, file, or directory). Section 2.2.1 describes how these types of access apply to disk and magnetic tape volumes; Section 2.2.2 describes how the access types apply to files and directories.

Note that CONTROL access is not explicitly set with the SET PROTECTION command. CONTROL access to a file is a function of ownership and privilege; therefore, it is never granted to the GROUP and WORLD categories of users, but is always granted to SYSTEM and OWNER.

You can bypass all UIC-based protection checks if you have the BYPASS user privilege. To bypass all protection checks and gain READ and CONTROL access to a disk file, you need READALL user privilege. For magnetic tapes, you can bypass UIC-based protection checks if you have VOLPRO (volume protection override) privilege.

For information on how to set UIC-based protection, see the discussion of protection codes in the *VMS DCL Concepts Manual*.

2.1.2 Access Control List (ACL)-Based Protection

An access control list (ACL) is primarily a list of entries that grant or deny access to a particular system resource, such as a disk file or a directory. Each access control list consists of one or more entries known as access control list entries (ACEs).

ACLs offer you the chance to "fine tune" the action taken when access to a file is sought. You can opt to provide an ACL on any disk file to permit as much or as little access as is desirable in each case. In providing a more detailed definition of who is allowed a particular kind of access, ACLs can enhance the security of disk files. Note, however, that "fine tuning" is done at the expense of performance; the larger the ACLs are, the more time they take to process.

ACL-based protection is supported for files and directories on disk volumes only. ACL protection is not supported for files on magnetic tape volumes.

If a file does not have an associated ACL, the system uses UIC-based protection to determine access as described in Section 2.1.1.

If a file has an associated ACL, the system uses the ACL to determine access as follows:

- If the ACL includes an identifier entry for a particular UIC, the ACL specification takes precedence over your UIC-based protection. However, SYSTEM and OWNER may still have access. For detailed information on ACL protection, see the *Guide to VMS System Security*.
- If the ACL does not include an identifier entry for your UIC (that is, does not explicitly allow or refuse access), then the system uses UIC-based protection to determine access.

If you have CONTROL access to a resource, you can define the access control list using the DCL commands EDIT/ACL, SET ACL and SET FILE/ACL.

For more information on how to invoke, modify, and display ACLs, see the *VMS Access Control List Editor Manual* or the *Guide to VMS System Security*.

2.1.3 VMS ANSI-Labeled Magnetic Tape Accessibility Protection

The VMS magnetic tape file system supports accessibility protection based on the ANSI and ISO standards. This protection scheme allows an installation to use a routine that is designed to interpret the contents of the volume- and header-label accessibility field. See the $MTACCESS system service in the *VMS System Services Reference Manual* for more information on installation routines.

When the installation routine is called to interpret a VOL1 label supplied as input, the routine must return one of the following values to the magnetic tape file system:

- User has no access to the volume or file
- User has complete access to the volume or file
- The output of any other VMS protection mechanism specified (for file-level access, the protection defaults to whatever was specified for the volume)

If the installation routine determines that you have no access to a volume or file, then you will be denied access regardless of the volume protection set by VMS. That is, no privilege granted to you by VMS can override that decision. On the other hand, if the installation determines that you have complete access to the file or volume, then you will be granted complete access even if a VMS protection scheme denies you access to that file or volume.

When the installation routine is called to return the VOL1 label or HDR1 label as output, the routine must return the character to write into the volume- or file-label accessibility field.

If you do not design your own installation routine, the VMS operating system provides a default routine for you, which works in the following way. The default installation routine first checks the ANSI standard version of the label. For magnetic tapes with a version number of 3 or less, the routine outputs either an ASCII character space (full access without checking VMS protection) or the character specified by the user. On input of these magnetic tapes, the routine checks for a blank (full access without checking VMS protection) and, if the field is not blank, returns the value that causes the file system to check for explicit override of accessibility checking.

For magnetic tapes with a version number greater than 3, the routine outputs either the character specified by the user or an ASCII 1 if no character was specified. On input of these magnetic tapes, the routine checks for an ASCII character space. If the field has an ASCII space, the user is given full access and VMS protection is not checked. If the field contains an ASCII 1 and the VMS protection has been specified, the VMS protection is checked.

If the field does not contain an ASCII space or an ASCII 1, the routine returns the value to the magnetic tape file system that forces the user to override the accessibility checking and allows the magnetic tape file system to check VMS protection.

When called to return a VOL1 or HDR1 label as output, the routine returns the ASCII character 1 if a VMS protection characteristic has been specified for the volume. If no VMS protection was specified, then the routine returns the ASCII character space.

When the installation routine is called to interpret a VOL1 or HDR1 label supplied as input, the routine returns the value that causes the magnetic tape file system to check the VMS protection under the following conditions:

- If the routine finds an ASCII 1 in the accessibility field when a VMS protection has been specified and the magnetic tape conforms to an ANSI standard greater than Version 3
- If the magnetic tape conforms to an ANSI standard greater than Version 3
- An ASCII space for all other cases

For all other conditions, the routine provided by VMS returns the value to the magnetic tape file system that causes the file system to check for explicit override of accessibility processing.

For more information on how ANSI-standard accessibility protection applies to files on magnetic tape volumes, see Section 2.2.2.4.

2.2 File Protection

This section discusses file protection at both the volume level and file level.

2.2.1 Volume-Level Protection

Disks and tapes can both be protected at the volume level. When you prepare a disk or magnetic tape volume for private use, you can define the protection you want applied.

Volume protection for a disk or magnetic tape volume is usually set when the volume is mounted or initialized. If you do not explicitly specify the protection for a particular disk volume, the system provides a default protection for you.

To change the protection that has been set on a disk volume, use the DCL command SET VOLUME as described in the *VMS DCL Dictionary*. For more information on setting the volume protection when you mount a volume, see the *VMS Mount Utility Manual* and the descriptions of the DCL command INITIALIZE in the *VMS DCL Dictionary*.

2.2.1.1 Disk Volume Protection

Disk volume protection supports the following access types:

- READ
- WRITE
- EXECUTE
- DELETE

If you have READ access to a disk volume, you have the right to examine, print, execute, or copy files from that volume.

If you have WRITE access to a disk volume, you have the right to modify existing files on that volume. Unlike that of magnetic tape volumes, WRITE access to a disk volume does not automatically imply READ access. That is, it is possible to grant a user WRITE access to a disk volume without granting that user READ access to the volume.

When applied at the volume (as opposed to the file) level, EXECUTE access means that you have the right to create files or modify existing files. When applied at the file level, EXECUTE access gives you the right to invoke executable files (such as executable images and command procedures).

If you have DELETE access to a disk volume, you have the right to delete files on the volume.

2.2.1.2 Protection of Disk Volumes

By default, no protection is applied to newly initialized disk volumes. You can specify protection with the /PROTECTION qualifier of the INITIALIZE command, and you can specify an ACL for a disk volume. The following example specifies UIC-based protection for the disk volume ACCOUNT1:

```
$ INITIALIZE WORKDISK: ACCOUNT1 -
_$ /PROTECTION=(S:RWED,O:RWED,G:R,W:R)
```

You can respecify the protection each time you mount the volume by using the /PROTECTION qualifier of the MOUNT command. You must own the volume or have VOLPRO privilege to change protection.

You can also limit access to a disk volume with the following qualifiers to the INITIALIZE and MOUNT commands:

- /SYSTEM — All processes have RWED access to the volume, but only system processes (or processes with SYSNAM and SYSPRV privileges) can create first-level directories. (The volume is owned by [1,1].)
- /GROUP — System, owner, and group processes have RWED access to the volume. World users have no access.
- /NOSHARE — System and owner processes have RWED access to the volume. Group and world users have no access.

At initialization time, the above qualifiers override any protection mask specified. At mount time, however, the protection mask overrides the qualifiers. When mounting or dismounting a volume, you must have GRPNAM privilege to specify /GROUP and SYSNAM privilege to specify /SYSTEM.

2.2.1.3 Magnetic Tape Volume Protection

Unlike disk volume protection, only READ and WRITE access apply to magnetic tape volume protection.

If you have READ access to a magnetic tape volume, you have the right to examine, print, or copy files from the volume. If you have WRITE access to a magnetic tape volume, you have the right to append or write files to the volume.

For magnetic tape volumes, WRITE access implies READ access. Thus, granting a category of users WRITE access to a magnetic tape volume automatically permits them to have READ access to the volume also.

There are two ways to protect a magnetic tape volume on a VMS system. You can protect a magnetic tape volume through the UIC-based protection scheme supported by VMS software. This scheme is checked on VMS systems only and will be ignored in any interchange with non-VMS systems. You can also protect a magnetic tape volume by using the guidelines of the ANSI standards. This protection scheme supports the protection of magnetic tape volumes in environments where interchange exists between VMS and non-VMS operating systems.

2.2.1.3.1 Protection for VMS Magnetic Tapes

The VMS magnetic tape file system provides two levels of protection. One level is defined by the ANSI standard and is encoded in the ACCESSIBILITY field of the first volume label written on the magnetic tape. The second level is UIC-based and is defined by the VMS magnetic tape file system. This protection is encoded in the second volume label written on the magnetic tape.

2.2.1.3.2 Protection for Interchange Environments

Magnetic tape volume protection is also supported for interchange between VMS and non-VMS operating systems. Protection is supported for interchange between VMS and DIGITAL operating systems other than VMS (such as RSX–11M and TOPS–20) as well as between VMS and non-DIGITAL operating systems.

For magnetic tapes processed on any operating system that supports a version of the ANSI standard later than Version 3, the accessibility information in the first volume label is processed exactly as described above. Magnetic tapes processed on operating systems other than VMS Version 4.0 or later have their protection characteristics encoded in the first volume label of the magnetic tape volume.

In order to process a magnetic tape created on a DIGITAL operating system other than VMS, a user must have VOLPRO privilege and must explicitly override the check on the protection. If the magnetic tape had been created with a specified accessibility, then a user wishing to access that tape must have the appropriate privilege and must explicitly override the check on accessibility. If the magnetic tape volume had not been created with such a protection scheme, then a user wishing to access that magnetic tape would be granted READ and WRITE access to that magnetic tape volume.

The VMS magnetic tape file system allows you to specify values for the fields in which other DIGITAL operating systems currently write their protection information. Except under conditions described in the two previous sections, VMS will not process this field. Thus, this field can be used to store the protection values for another operating system without affecting the VMS protection characteristics on that particular volume.

2.2.2 File-Level Protection

In addition to volume-level protection, the VMS operating system supports protection at the file level for all files residing on disk volumes, including directories. In certain cases, file-level protection is also supported for magnetic tape files. For files residing on magnetic tape volumes, however, only one special case (described in Section 2.2.2.4) exists in which distinct file-level protection applies.

File-level protection is described as it applies to each of the following:

- Disk file protection
- Directory file protection
- Magnetic tape file protection

2.2.2.1 File Protection

For the most part, file protection is transparent. The tools exist, however, to adjust the protection of a file as you see fit. You must own the file; have control access to the file; or have GRPPRV, SYSPRV, BYPASS, or READALL privilege to set the protection or modify the ACL of a file.

Note: You cannot completely protect a file without applying at least the same protection to the directory in which the file resides; see Section 2.2.2.3.1 for information on directory protection.

2.2.2.1.1 Default File Protection

A new file receives default UIC-based protection and the default access control entries (ACEs), if any, of its parent directory. A renamed file's protection is unchanged. A new version of an existing file receives the UIC-based protection and ACL of the previous version. (Use the /PROTECTION qualifier of the BACKUP, COPY, and CREATE commands to override the default UIC-based protection.)

- Default UIC protection — The operating system provides each process with a default UIC-based protection of (S:RWED,O:RWED,G:RE,W). To change the default protection, enter the SET PROTECTION command with the /DEFAULT qualifier. For example, if you place the following command in your login command procedure, you grant all processes read and execute access to any files that you subsequently create:

  ```
  $ SET PROTECTION = (S:RWED,O:RWED,G:RE,W:RE)/DEFAULT
  ```

  (Remember that you must execute the login command procedure for this command to execute.)

- Default ACL protection — You can override default UIC protection for specified directories or subdirectories by placing a default-protection ACE in the ACL of the appropriate directory file. The default protection specified in the ACE is applied to any new file created in the specified directory or in any subdirectory of the directory. The following ACE, which must be in the ACL of a directory file, specifies that the default protection — for that directory and for the directory's subdirectories — allows system and owner processes full access, group processes read and execute access, and world users no access:

  ```
  (DEFAULT_PROTECTION,S:RWED,O:RWED,G:RE,W:)
  ```

To specify a default identifier ACE to be copied to the ACL of any file subsequently created in the directory, specify the DEFAULT option in the directory file's identifier ACL.

2.2.2.1.2 Explicit File Protection

You can explicitly specify UIC-based protection for a new file with the /PROTECTION qualifier (valid with the BACKUP, COPY, RENAME, and CREATE command), as demonstrated in the following command line:

```
$ CREATE MAST12.TXT/PROTECTION=(S:RWED,O:RWED,G,W)
```

You can change the UIC-based protection on an existing file with the SET PROTECTION command, as follows:

```
$ SET PROTECTION=(S:RWED,O:RWED,G,W) MAST12.TXT
```

After a file is created and you have created an ACL for the file, you can modify the ACL and add as many ACEs to the ACL as you desire. The protection specified by the ACL overrides the file's UIC protection.

2.2.2.2 Disk File Protection

Each file on a disk has its own protection code, which is distinct from the protection that applies to the disk volume itself. For files residing on disk volumes, the following access types are supported:

- READ
- WRITE
- EXECUTE
- DELETE
- CONTROL

If you have READ access to a file or group of files on a disk volume, you have the right to examine, print, or copy that file or group of files. READ access automatically includes EXECUTE access to a specified file or group of files on disk.

If you have WRITE access to a file or group of files on a disk volume, you have the right to modify the file or group of files. As with disk volume protection, it is possible to be granted WRITE access without having READ access. This is not very useful, however, since opening a file for a write operation implies READ access.

If you have EXECUTE access to a disk file or group of disk files, you have the right to execute executable program images or DCL command procedures contained in that file or group of files.

If you have DELETE access to a particular disk file or group of disk files, you have the right to delete the file or group of files.

If you have CONTROL access to a particular disk file or group of disk files, you have the right to change the characteristics of the file or group of files.

You can specify a protection code when you create or copy a file by using the /PROTECTION qualifier, as in the following example:

```
$ COPY/PROTECTION=(SYSTEM:RW,OWNER:RWED,GROUP:RW,WORLD) ABC.DAT
```

You can also change the protection for an existing file by using the SET PROTECTION command. For example, enter the following:

```
$ SET PROTECTION=(SYSTEM:RWE,OWNER:RWED,GROUP:RE,WORLD) ABC.DAT
```

This command gives the VMS operating system the following instructions regarding the file ABC.DAT: SYSTEM has READ, WRITE, and EXECUTE privileges; OWNER has READ, WRITE, EXECUTE, and DELETE privileges; GROUP has READ and EXECUTE privileges only; and WORLD has no access.

For SYSTEM and OWNER, CONTROL access is implied and unchangeable, but not so for GROUP and WORLD.

If you do not define a protection code for a file when creating it, the system applies a default protection. If a version of the file already exists, protection is taken from the previous version. For a new file, protection is determined in one of the two following ways:

- If the directory where the file is to be catalogued has an associated access control entry that specifies the DEFAULT_PROTECTION entry, then the specified protection is used.
- If the directory does not have the DEFAULT_PROTECTION entry, then the default process protection is used. The default process protection is established explicitly with the SET PROTECTION/DEFAULT command, or by default when you log in.

Show the current default protection by entering the SHOW PROTECTION command as follows:

```
$ SHOW PROTECTION
SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=NO ACCESS
```

This response is the system default protection. It indicates that SYSTEM and OWNER have all types of access, members of the owner's group have READ and EXECUTE access, and all other users (WORLD) have no access.

To determine the current protection associated with a specific file or files, use the /PROTECTION qualifier with the DCL command DIRECTORY, as in the following example:

```
$ DIRECTORY/PROTECTION  MYFILE.REC
Directory DBA1:[CRAMER]

MYFILE.REC;5        (RWED,RWED,RW,R)

Total of 1 file.
```

You can change the default protection applied to files created during an interactive session by using the SET PROTECTION/DEFAULT command. The SET PROTECTION/DEFAULT command indicates that the protection code you specify is to be applied to all files subsequently created during the terminal session or batch job (providing that the files are not subject to other sources of protection).

Note that, to completely protect a file, you must protect both the file itself and the directory in which the file is listed. If you have files that must be protected against unauthorized access, be sure to specify the proper protection both for the directories in which the files are listed and for the files themselves.

2.2.2.3 Directory File Protection

Each directory file has a protection associated with it. The directory protection can override the protection of individual files within the directory. For example, if a directory denies WORLD access, WORLD users cannot look up even those files in the directory that permit WORLD access.

For directory file protection, the following access types are supported:

- READ
- WRITE
- EXECUTE
- DELETE
- CONTROL

Having READ access to a file in a directory means that you have the right to examine, print, or copy that file. If you have READ access to a directory file, you can display the contents of the directory file with the DIRECTORY command. For example, if you have access to the directory [JONES], you can enter the following:

```
$ DIRECTORY [JONES]
```

This command generates a display of the files contained in the [JONES] directory.

If you have READ access, you can access any file listed in the directory, unless the protection on that file denies you access. If the protection applied to the whole directory denies you READ access, then you cannot access even those files in the directory that permit access to users in your group.

If you have WRITE access to a directory file, you have the right to modify or write to that directory file. However, you must have both READ and WRITE access to a directory in order to create files in that directory, to rename files in that directory, or to perform any file operation that involves changes to that directory file.

Note that EXECUTE access has a special meaning when applied to directories. EXECUTE-only access to a directory file allows you access only to files that you can identify by name. It also means that you can access files in the directory that are not protected against users in your category, provided that you do not perform an operation that modifies the directory file. However, you cannot list all the entries in a directory by using wildcards.

For example, assume you have EXECUTE-only access to the [JONES] directory, and you enter the following command:

```
$ DIRECTORY [JONES]
```

The system responds with an error message and does not list the files in the [JONES] directory. However, if you know that the file DATAFILE.DAT resides in the [JONES] directory, you can enter the following command:

```
$ TYPE [JONES]DATAFILE.DAT
```

This causes the system to display the contents of the file. Thus, EXECUTE-only access provides some, but not all, of the operations that READ access provides.

If you have DELETE access to a directory file, you have the right to delete that directory file. You must remove all entries from a directory file before you can delete it. When you create a directory file with the CREATE/DIRECTORY command, you do not, by default, get DELETE access. If you want to be able to delete a directory file, you must use the DCL command SET PROTECTION to explicitly assign DELETE access to the OWNER category.

If you have CONTROL access to a directory file, you have the right to change the characteristics of the directory file.

Remember that, to ensure that your disk files are adequately protected, you must make certain they are properly protected at both the directory and file levels. ACLs allow you to further control access to disk files. For more information on ACLs, see the *VMS Access Control List Editor Manual*.

2.2.2.3.1 UIC Directory Protection

You cannot completely protect a file without applying at least the same protection to the directory in which the file resides. For example, if you deny a user all access to a file but allow that user READ access to the file's directory, the user cannot access the contents of the file but can see that it exists. Conversely, a user allowed access to a file and denied access to the file's directory (or one of the parent directories) cannot see that the file exists.

Note: To protect sensitive files, the directory protection alone is not adequate. You must also protect each individual file contained within the directory.

By default, top-level directories receive UIC-based protection (S:RWE,O:RWE,G:RE,W:E) and no ACL. Subdirectories receive UIC-based protection minus any DELETE access or default protection ACEs from the parent directory.

To specify UIC-based protection explicitly when creating a directory, use the /PROTECTION qualifier of the CREATE/DIRECTORY command. You cannot specify an ACL for the directory until the directory is created. To change the UIC-based protection of an existing directory, use the SET PROTECTION command (apply this command to the directory file). To specify or change the ACL of an existing directory, edit the directory file's ACL (see Section 2.2.2.1.1).

You can limit but not prohibit directory access by specifying EXECUTE access but not READ access. EXECUTE access on a directory permits you to examine and read files that you know are contained in the directory (that is, you know the file specifications) but prevents you from displaying a list of the files in the directory.

2.2.2.4 Magnetic Tape File Protection

In general, the protection that applies to a magnetic tape volume also applies to all the files on that volume. For VMS ANSI-labeled magnetic tapes, however, distinct file-level protection is also supported. For VMS ANSI-labeled magnetic tapes, file-level protection can be determined by the contents of the accessibility field, which is located in the first header label of each file.

Each time a VMS ANSI-labeled magnetic tape file is opened for processing, the installation routine examines the contents of the accessibility field of that file. Depending on the value returned by this routine, access to the file in question will either be granted, denied, or defaulted to the VMS protection scheme applied to the volume containing that file (see Section 2.1.3).

2.2.2.5 Protection of Mail Files

Note: (Requires the Common Utilities Option.)

Mail files receive the protection (S:RW,O:RW,G,W). Files of type MAI created with the Mail Utility EXTRACT/MAIL command receive the protection (S:RW,O:RW,G,W).

2.2.2.6 Displays of Ownership and Protection

You can display ownership and protection information with the following commands and qualifiers:

Command	Display
DIRECTORY/ACL filespec	File's ACL
DIRECTORY/OWNER_UIC filespec	File's UIC
DIRECTORY/PROTECTION filespec	File's UIC-based protection
DIRECTORY/SECURITY	All of the above
DIRECTORY/FULL filespec	All of the above and more
SHOW ACL obj-name	Device, file, logical name table, or global section's ACL
SHOW PROCESS	Process UIC
SHOW PROTECTION	Default file protection
SHOW DEVICES/FULL device-name	Device UIC and protection

2.3 Device Protection

In general, device protection controls the ability to allocate the device and is specified by granting READ access in an ACL. To specify an ACL for a disk device, use the SET ACL/OBJECT_TYPE=DEVICE command. For example, to grant READ access to the disk device WORKDISK to a user with the alphanumeric UIC [FRED], enter the following:

```
$ SET ACL/OBJECT_TYPE=DEVICE/ACL=(IDENTIFIER=[FRED],ACCESS=READ) -
_$ WORKDISK
```

Note that, when you mount an ACL for a disk device, the ACL is associated with the device not with the disk volume. For example, if you mount a disk device on WORK1 and specify the preceding SET ACL/OBJECT_TYPE=DEVICE command and then dismount the disk device, the ACL protection remains on WORK1 but not on the disk device.

The only protection that applies to a nonfile device is the ability to allocate it, specified by READ access. By default, nonfile devices such as mailboxes are unprotected. Interactive terminals are set up to provide complete access to system users and no access to all other users. (Note that access here refers to access via an application program. The device protection on a terminal does not control who can log in on it.) You can change the protection of a nonfile device through the use of ACLs or by changing the standard UIC

protection. Modify the ACL with the DCL command SET ACL/OBJECT_TYPE=DEVICE. Modify the UIC protection with the DCL command SET PROTECTION/DEVICE (requires OPER privilege). For example, the following command allows users holding the PAYROLL identifier to have READ access to TERMINAL3:

```
$ SET ACL/OBJECT_TYPE=DEVICE/ACL=(IDENTIFIER=PAYROLL,ACCESS=READ)-
_$ TERMINAL3
```

3 Preparing Volumes for Private Use

You can prepare private disk and magnetic tape volumes for routine operations by using the DCL commands ALLOCATE, INITIALIZE, and MOUNT. Each of these DCL commands is discussed in the sections that follow.

If you are interested in setting up public volumes, see the description in the *Guide to Setting Up a VMS System*.

3.1 Setting Up a Private Volume

Under some circumstances, it may be desirable to perform your work on a device that cannot be accessed by unauthorized users. By creating a private volume and mounting it on a device allocated exclusively to your process, you can perform your work without fear of interference from other users.

To set up your private volume, perform the following steps:

1. Use the *ALLOCATE* command to assign a disk drive or magnetic tape drive exclusively to your process.
2. Use the *INITIALIZE* command to format the volume and write an identifying label on the volume.
3. Use the *MOUNT* command to make a volume, and the files or data it contains, accessible to your process.

See the section dealing with the individual DCL command for a complete description of each parameter and qualifier. Also see the DCL commands ALLOCATE and INITIALIZE in the *VMS DCL Dictionary*, and for more information on MOUNT, see the *VMS Mount Utility Manual*.

3.2 Allocating Disks and Magnetic Tape Drives to Your Process

Before you can begin processing files or data on a private volume, you must first allocate a drive to your process. Use the DCL command ALLOCATE to logically assign a disk drive or a magnetic tape drive to your process.

Whether you are allocating a disk drive or a magnetic tape drive, the format for the ALLOCATE command is as follows:

```
ALLOCATE device-name[:] [logical-name]
```

Command Parameters

device-name[:]

Specifies the drive on which the volume will be loaded. The device name can be a physical, generic, or logical name. A physical device name consists of a device code, alphabetic controller designation, and a unit number. A generic device name consists only of the device code. A logical name must equate to a physical or generic name. Use of the colon is optional but recommended by DIGITAL.

[logical-name]
Specifies an optional logical name to be associated with the specified disk or magnetic tape drive.

The ALLOCATE command allocates only one device to a process. Except for a list of generic device names, ALLOCATE does not accept lists of device names in the command string. Although you can specify a list of generic devices, the first available device will be the only one to be allocated. For a list of available devices, refer to the Software Product Description (SPD) supplied with your system software.

In the examples that follow, the first two show how to use the ALLOCATE command to allocate disk drives, while the next three show how to allocate magnetic tape drives.

Examples

1
```
$ ALLOCATE DM: DISK
```

In this example, the ALLOCATE command requests that the first available RK06 or RK07 be assigned to your process. The logical name DISK is placed in your process logical name table and assigned the name of the allocated device. Other users are unable to access the device.

2
```
$ ALLOCATE DMB2:
%DCL-I-ALLOC, _MARS$DMB2: allocated
```

In this example, the ALLOCATE command specifies a physical device named DMB2, which requests the allocation of a specific RK06 or RK07 disk drive; that is, unit 2 on controller B. The response from the ALLOCATE command indicates that the device was successfully allocated.

If you want to allocate a particular type of device, use the /GENERIC qualifier with the ALLOCATE command. For example, device DM could be an RK06 or RK07 disk. If you specifically want to allocate an RK07, use the /GENERIC qualifier in the following way:

```
$ ALLOCATE/GENERIC RK07  MYDISK
```

In this case, the system allocates the first available RK07 device to your process. For further discussion of the /GENERIC qualifier as well as all of the qualifiers applicable to the ALLOCATE command, see the *VMS DCL Dictionary*.

3
```
$ ALLOCATE MTA1:
%DCL-I-ALLOC, _MARS$MTA1: allocated
```

In this example, the ALLOCATE command specifies a physical device named MTA1.

The operating system informs you that MTA1 has been allocated.

```
4  $ ALLOCATE MF,MT,MS DRIVE
   %DCL-I-ALLOC, _MARS$MTA0: allocated
```

In this example, the ALLOCATE command specifies a list of generic device names. At a minimum, a generic device name consists of the device code; a controller designation is optional. Only one of the specified generic devices is allocated. Each element in the list must represent a unique generic device type.

The VMS operating system informs you that drive MTA0 has been allocated. Although it is not indicated in the message, the system also assigns the logical name DRIVE to the drive MTA0.

```
5  $ ALLOCATE DRIVE1: D1
   %DCL-I-ALLOC, _MARS$DBA3: allocated
```

In this example, the ALLOCATE command specifies a logical name, DRIVE1, as the device name and assigns a new logical name, D1. (This example assumes that DRIVE1 has already been defined as the physical device DBA3.)

The VMS operating system informs you that DRIVE1 has been allocated. Although it is not indicated in the message, VMS also assigns the new logical name D1 to the drive DRIVE1.

If you want to allocate a specific type of magnetic tape device, use the /GENERIC qualifier. For example, if you want to allocate a TU78 device specifically, you would use the /GENERIC qualifier with the ALLOCATE command, as follows:

```
$ ALLOCATE/GENERIC TU78  TAPE_TU78
```

In this case, the system would allocate the first available TU78 device to your process. For a further discussion of the /GENERIC qualifier as well as all of the qualifiers applicable to the ALLOCATE command, see the *VMS DCL Dictionary*.

3.3 Initializing a Volume

Before you can write files or data to a disk or magnetic tape volume, the volume must be initialized. You can use the DCL command INITIALIZE to format and write a label to the volume.

The INITIALIZE command does the following:

- Invalidates all existing data on the volume, if any, and creates a new file structure
- Writes a label on the volume to identify it
- Defines the owner UIC and the protection for the volume

Whether you are initializing a disk or magnetic tape volume, the format for the INITIALIZE command is as follows:

```
INITIALIZE  device-name[:]  volume-label
```

Command Parameters

device-name[:]
Specifies the name of the device on which the volume is to be physically mounted and then initialized. To prevent initializing another user's volume, you should allocate a device before you initialize the volume. Prior allocation is not required, however.

volume-label
Specifies the identification to be encoded on the volume. You can specify up to 12 alphanumeric characters for a disk volume, or up to 6 alphanumeric characters for a magnetic tape volume. Alphabetic characters are automatically changed to uppercase. The first character of a disk volume-label specification must be alphanumeric.

There are some cases where you might be prevented from accessing and initializing a particular volume. For example, if the volume that you want to initialize previously contained data, the protection code may prevent you from accessing and initializing that particular volume. In the case of a magnetic tape volume, you may not be able to initialize the volume if the first file on the volume has not reached its expiration date or if the volume or file accessibility is such that the installation routine provided by the VMS operating system does not allow you to access the volume.

If the volume is protected or if the expiration date on the first file has not been reached and you are not the owner or a SYSTEM user, you must have VOLPRO user privilege to override volume protection. If you do not have VOLPRO privilege, you can ask the previous owner of the volume or another user who does have READ/WRITE access (the system manager or operator, for example) to initialize it for you. If the installation routine provided by the VMS operating system (or by a user-designed installation routine) does not allow you access, then consult your system manager.

When you give the volume to another user for initialization, you should specify the following:

- The label you want to have written on the volume
- The protection code and owner UIC you want assigned to the volume

When you obtain a magnetic tape or disk volume, place identification on the outside of the volume so that it can be easily identified.

The next two sections describe how to initialize disk and magnetic tape volumes.

3.3.1 Initializing a Disk Volume

By default, the INITIALIZE command builds a Files–11 structure on your new volume. The default format for disk volumes in the VMS operating system is called the Files–11 On-Disk Structure Level 2. The INITIALIZE command can also initialize disk volumes in the Files–11 On-Disk Structure Level 1 format.

You do not need special privileges to override logical protection on a blank disk volume (that is, a volume that has never been written to) or on a disk volume that is owned by your current UIC or by UIC [0,0]. In all other cases, you must have the user privilege VOLPRO to initialize a disk volume.

The following examples include typical cases of initializing a disk.

Examples

1
```
$ INITIALIZE DISK USER_DISK
```

In this example, the volume is given the label USER_DISK. DISK is the logical name of the device on which it is mounted.

2
```
$ ALLOCATE DJA2:  TEMP
%DCL-I-ALLOC, _MARS$DJA2 allocated
$ INITIALIZE  TEMP:  BACKUP_FILE
```

This example shows how to initialize an RA60 volume. First, the drive is allocated to ensure that no one else can access it. Then, when the volume is physically loaded on the drive, the INITIALIZE command initializes it. Refer to the Software Product Description (SPD) for a complete list of devices that are available.

3.3.2 Initializing a Magnetic Tape Volume

Use the INITIALIZE command to initialize a magnetic tape volume. The default format for magnetic tape volumes in the VMS operating system is based on Level 3 of the ANSI and ISO standards for magnetic tape labels and file structure for informational interchange).

Use the DCL command INITIALIZE to encode VMS ANSI-labeled format on a magnetic tape volume. INITIALIZE writes labels to an empty file on the magnetic tape volume in the following order:

1. A volume label
2. File-header labels with the file sequence number set to 0
3. Two tape marks framing an empty file (BOT and EOT)
4. Corresponding end-of-file labels (EOF)
5. A double tape mark, specifying logical end-of-volume

The following example describes how to initialize a magnetic tape volume:

```
$ INITIALIZE TAPE USER
```

In this example, the magnetic tape volume is given the label USER. TAPE is the logical name of the device on which it is mounted.

Note that, if you use ANSI "a" characters (which are not alphanumeric) on the volume label on magnetic tape, you must enclose the volume name in quotation marks.

3.4 Mounting a Volume

Before you can begin processing files or data on your private disk or magnetic tape volume, make sure the volume is mounted. Use the DCL command MOUNT to make a disk or magnetic tape volume and the files or data it contains accessible to your process.

When you enter the MOUNT command, the system checks the following:

- That the device has not been allocated by another user
- That the device protection allows you to allocate the device
- That a volume is physically loaded on the device specified
- That the label on the volume matches the label specified

You can mount a single volume or a volume set. Binding volumes into a volume set allows you to extend the space available for your files by adding volumes to the set, rather than defining new volumes.

The procedures for creating and mounting disk volume sets and magnetic tape volume sets (as opposed to single volumes) are described in the sections that follow. Whether you are mounting a disk or a magnetic tape volume, the format for entering the MOUNT command is as follows:

```
$ MOUNT device-name[:][,...] [volume-label[,...]] [logical-name[:]]
```

Command Parameters

device-name[:][,...]
Specifies the physical device name or logical name of the device on which the volume is to be mounted. If you specify more than one device name for a disk or magnetic tape volume set, separate the device names with either commas (,) or plus signs (+).

volume-label[,...]
Specifies the label on the volume. If you specify more than one volume label, separate the labels with either commas or plus signs. The volumes must be in the same volume set. For magnetic tape volumes, the labels must be specified in ascending order according to relative volume number.

The volume-label parameter is not required when you mount a volume with the /FOREIGN or /NOLABEL qualifier or when you specify /OVERRIDE=IDENTIFICATION. To specify a logical name when you enter either of these qualifiers, type any alphanumeric characters in the volume-label parameter position.

logical-name[:]
Defines a 1 through 255 alphanumeric character string to be associated with the device.

If you do not specify a logical name for a disk drive, the MOUNT command assigns the default logical name DISK$volume-label to individual disk drives; it assigns the default logical name DISK$volume-set-name to the device on which the root volume of a disk volume set is mounted. Similarly, if you do not specify a logical name for a magnetic tape drive, the MOUNT command assigns only one logical name, TAPE$volume-label, to the first magnetic tape device in the list. For a volume set, no logical name is assigned unless you specify one.

The MOUNT command places the name in the job logical name table, unless you specify /GROUP or /SYSTEM. In the latter cases, it places the logical names in the group or system logical name table. You should avoid assigning a logical name that matches the file name of an executable image in SYS$SYSTEM. Such an assignment will prohibit you from invoking that image.

At many installations, operators perform the physical mounting (and dismounting) of both system and private disk and magnetic tape volumes. Since operators at such installations assist you in your MOUNT requests, you do not need to include the /ASSIST qualifier with the MOUNT command. For example, the following command notifies the operator of your mount request and displays a message at your terminal:

```
$ MOUNT DMA1: DISK VOL1
%MOUNT-I-OPRQST, PLEASE MOUNT DEVICE _MARS$DMA1:
```

After the device has been successfully mounted, you are notified with the following message:

```
%MOUNT-I-MOUNTED, DISK mounted on _DMA1:
```

As an alternative to requesting a specific device, you might want to request a device type. In the following example, MOUNT allocates an available device of the specified type and requests operator assistance in mounting it:

```
$ MOUNT DB: USER_DISK DISK
%MOUNT-I-OPRQST, Please mount volume USER_DISK in device _NODE$DBA0:
%MOUNT-I-MOUNTED, USER_DISK        mounted on _NODE$DBA0:
%MOUNT-I-RQSTDON, operator request canceled - mount completed successfully
```

DISK is the logical name created when the RP06 disk unit is allocated. Since the device is allocated to your process, no other user can access the volume. Your access to USER_DISK is determined by the volume protection code and the volume UIC.

Operator assist messages are sent to all operators enabled to receive TAPE and DISK messages. Thus, if operator assistance is needed for mounting a disk device, a message is sent to disk operators.

Any operator reply to a MOUNT request is written to SYS$OUTPUT to be displayed on the user's terminal or written in a batch job log.

If no operator is available (operator is not enabled) to receive and respond to a MOUNT request, a message is displayed to inform you of the situation. A volume placed in the requested drive needs no additional operator assistance. Note that you can specify the /NOASSIST qualifier to avoid operator assistance.

The following sections provide examples of mounting volumes; however, they do not all include operator assistance messages.

3.4.1 Mounting a Disk Volume

Use the following procedure to allocate, initialize, and mount a disk volume:

```
$ ALLOCATE DMA2:  TEMP
%DCL-I-ALLOC, _MARS$DMA2: allocated
$ INITIALIZE  TEMP:  BACKUP_FILE
$ MOUNT  TEMP:  BACKUP_FILE
%MOUNT-I-MOUNTED, BACKUP_FILE mounted on _DMA2:
```

If you want to mount a foreign disk volume (that is, one having a file structure other than Files–11), use the /FOREIGN qualifier in conjunction with the MOUNT command. The MOUNT/FOREIGN command makes the contents of your volume available to the system but makes no assumptions concerning its file structure. In the following case, assume that DISK is the logical name assigned to the device at the time of disk allocation:

```
$ MOUNT/FOREIGN DISK
%MOUNT-I-MOUNTED, BACKUP_FILE      mounted on DISK$DMA2:
```

Note that MOUNT reports a volume label even though the disk is mounted as a foreign device. MOUNT reports the label because the disk has a Files–11 structure; if a disk does not have a recognized file structure, no label is reported.

You need the user privilege VOLPRO to mount a Files–11 structured disk with the /FOREIGN qualifier, unless its owner UIC matches your own.

You must use the /FOREIGN qualifier if you want to use the VMS Bad Block Locator Utility (BAD) to locate and record bad blocks on your disk volume. BAD is useful for media preparation and is thus distinct from the volume preparation tasks described in this chapter. To invoke BAD, use the DCL command ANALYZE/MEDIA. For more information on BAD, see the description in the *VMS Bad Block Locator Utility Manual*

In addition to the /FOREIGN qualifier, many other qualifiers are supported for the MOUNT command. The /SYSTEM qualifier is described in the *Guide to Setting Up a VMS System*. The /BIND qualifier is discussed in the next section, which describes how to mount disk volume sets. For a complete list of the qualifiers, see the DCL MOUNT command.

3.4.2 Mounting a Disk Volume Set

When you mount a disk volume set, the volume label specified in the list must correspond to a device name in the same position in the device name list.

Two or more disk volumes can be bound into a volume set. The first volume in the set is called the *root volume*. Each volume in the set is identified by a volume number relative to the root volume, which is always relative to volume 1.

A disk volume set has a single directory structure. The master file directory (MFD) for the entire volume set resides on the root volume, which is always the first volume in the set.

When a disk volume set is on line and mounted, all files and directories in the set can be accessed by specifying either of the following:

- Device name of the device on which the root volume is mounted

- Logical name assigned to the volume set when it was mounted

Use the /BIND qualifier to create a disk volume set. When you use the MOUNT command with the /BIND qualifier to create a disk volume set, the /BIND qualifier identifies a volume set by assigning it a volume set name that applies to all volumes in the set. It also identifies the root volume and creates the directory structure for the volume.

When you create files on a volume set, the file system allocates space for the files anywhere on the set, wherever the most space exists. When existing files on any volume are extended, extension occurs on the same volume, unless the volume is physically full. You can add new volumes to a volume set whenever additional space is needed.

For example, all disk volumes that are mounted on a daily basis can be bound into a volume set. Since this set contains all user file directories, users do not need to specify device names in file specifications to access files on any volume in the volume set. In fact, the physical location of a file is of no concern to users of the system.

Note: Do not bind your system disk into a volume set. Volume sets are not supported by VMS software updates and optional product installation. If certain system files move or extend to other volumes in the set, the system may fail to boot.

No special privileges are required to use volume sets. However, you must have WRITE access to the index file on all volumes you are attempting to bind into a volume set; this usually means you also must have a system UIC, have the user privilege SYSPRV, or be the owner of the volumes.

You can create a disk volume set from newly initialized volumes, or you can create a volume set by extending an existing volume that already contains a directory structure and files. You can also add volumes to an existing volume set. The next three sections contain examples of how to create and mount each type of disk volume set.

3.4.2.1 Creating a Disk Volume Set from New Volumes

The following steps show how to create a disk volume set from new disk volumes. This example assumes there are no files or data on the volumes to be bound.

1. Allocate the necessary devices and physically load the volumes.
2. Initialize each volume in the set by entering the following:

```
$ INITIALIZE DB1:  PAYVOL1
$ INITIALIZE DB2:  PAYVOL2
$ INITIALIZE DB3:  PAYVOL3
```

   When you initialize volumes for a volume set, you can also use other qualifiers with the INITIALIZE command to define the volume ownership and protection. Protection and ownership information is obtained from the root (first) volume. The protection and ownership of the other volumes is ignored.

3. Enter the following MOUNT/BIND command to create the volume set:

```
$ MOUNT/BIND=MASTER_SET -
_$ DB1:, DB2:, DB3:   PAYVOL1, PAYVOL2, PAYVOL3
```

This MOUNT/BIND command defines the volume set name, MASTER_SET, and defines the relative volume numbers of the volumes PAYVOL1, PAYVOL2, and PAYVOL3.

A disk volume set name can have from 1 to 12 alphanumeric characters. The volume set name must be different from all volume labels within the set, and all labels in the set must be unique.

The order of the device names corresponds to the volume labels specified: PAYVOL1 must be physically loaded on DB1, PAYVOL2 on DB2, and PAYVOL3 on DB3.

PAYVOL1, which is listed first in the list of labels, becomes the root volume of the set. The master file directory (MFD) for PAYVOL1 contains the directory structure for the entire volume set.

Note that the MOUNT/BIND command creates the volume set and mounts the volumes. When this command completes successfully, all volumes in the set are ready for use — user file directories can be created.

The /BIND qualifier must be used only once to create the volume set. In subsequent use, the volume set may be mounted with a single MOUNT command. The following example illustrates the use of one MOUNT command to mount a previously created volume set:

```
$ MOUNT DB1,DB2,DB3 PAYVOL1,PAYVOL2,PAYVOL3
```

3.4.2.2 Creating a Disk Volume Set from an Existing Volume

The following example shows how to create a disk volume set from an existing volume, assuming that the volume USERFILES already contains a directory structure and files and that the volume is currently located on the device DM1:

```
$ DISMOUNT/NOUNLOAD DM1:

$ INITIALIZE DM2: USERFILES2
$ MOUNT/BIND=USERS -
_$ DM1:, DM2:    USERFILES, USERFILES2
```

The initial volume USERFILES must be specified first; it becomes the root volume of the set. When you create a volume set from an existing volume, you must specify that volume first because the file system must build on the existing directory structure.

Note that, if you attempt to create a volume set from two or more volumes that already contain files and data, the file system does not issue an error message when you enter the MOUNT/BIND command. However, the volumes are unusable as a volume set because the directory structures are not properly bound.

3.4.2.3 Adding Volumes to a Disk Volume Set

This section describes how to add volumes to an existing volume set. The following example assumes that the volume set named MASTER_SET is on line and mounted and has volumes named PAYVOL1, PAYVOL2, and PAYVOL3:

```
$ INITIALIZE DB4: PAYVOL4
$ MOUNT/BIND=MASTER_SET DB4: PAYVOL4
```

The MOUNT command binds the volume PAYVOL4 with the existing volume set and makes the volume ready and available for use. Note that, if the volume set MASTER_PAY was mounted with the /SYSTEM, /GROUP, or /SHARE qualifier, the MOUNT/BIND command that adds a volume to the set must also specify the appropriate qualifier.

When you add a volume to an existing set, the only volume in the set that must be mounted is the root volume, relative volume 1. None of the other volumes need be mounted.

You can also add a volume to a set at the same time you mount the set. The following procedure assumes an existing volume set named MASTER_SET with volumes named PAYVOL1, PAYVOL2, and PAYVOL3:

```
$ INITIALIZE DB4:  PAYVOL4
$ MOUNT/BIND=MASTER_SET  DB1:, DB2:, DB3:, DB4:
_$PAYVOL1, PAYVOL2, PAYVOL3, PAYVOL4/SYSTEM
```

Note that the first device/volume pair listed in the MOUNT/BIND command is the root volume of the set. When you add a volume to a set while mounting the set, you must list the root volume first.

Note: Once a volume is bound into a volume set, it cannot be "unbound."

You can add volumes to an existing volume set at any time. The maximum number of volumes in a set is 255.

3.4.3 Mounting a Magnetic Tape Volume

When mounting a magnetic tape volume, specify, along with the MOUNT command, the following: any qualifiers you choose; a device name; and, optionally, a label and logical name.

For a discussion of how to mount magnetic tape volume sets, see Section 3.4.4.

The next two sections describe procedures and commands for mounting single-volume, ANSI-labeled, magnetic tapes.

3.4.3.1 Mounting an ANSI-Labeled Volume

When you use the MOUNT command to mount a magnetic tape volume, VMS software checks to see whether the volume has a VMS or a non-VMS ANSI-labeled format. If the format is ANSI-labeled, the following are checked:

- The volume identifier field
- The protection on the ANSI-labeled volume as described in Section 2.1.3.

Mount an ANSI-labeled volume by including the device name and volume identifier as follows (specifying a logical name is optional):

```
$ MOUNT  MT:  ELAINE ET
%MOUNT-I-OPRQST, please mount volume ELAINE in device $MTA1:
%MOUNT-I-MOUNTED, ELAINE mounted on MTA1:
%MOUNT-I-RQSTDON, operator request canceled -- mount completed successfully
```

MOUNT finds an available MT drive, MTA1, and requests operator assistance. The message displayed at the user terminal indicates which drive has been selected. At this point, you (or the operator) load the magnetic tape on the drive, and the mount operation completes. No operator response is necessary. The display informs you that the volume named ELAINE is mounted on the drive MTA1. Although not indicated in the message, MOUNT also assigns the logical name ET to the volume ELAINE.

When used with the MOUNT command, the qualifiers described in the next section affect the label format of a volume or of the magnetic tape file system used to process an ANSI-labeled volume, or both. Unless otherwise noted, you must have VOLPRO privilege to use any of these qualifiers when the volume is a VMS ANSI-labeled volume containing protection that restricts your process from accessing the volume.

3.4.3.2 Using MOUNT Command Qualifiers

This section describes some of the command qualifiers you can use when mounting a magnetic tape volume. For a complete list of all the command qualifiers supported for the mounting of magnetic tape volumes, see the *VMS Mount Utility Manual*.

/BLOCKSIZE=n

Use the /BLOCKSIZE=n qualifier to specify the block size for the magnetic tape. The range of valid values for *n* varies and depends on the density of the volume, whether the data is for input or output, and whether the operation uses RMS.

By default, VMS writes 2048-byte blocks, which conforms to the ANSI standard. Although the VMS operating system allows you to specify a block size larger than 2048 bytes, a larger block size does not conform to ANSI standards.

You must specify /BLOCKSIZE when mounting volumes that do not support the second file header label as defined in the ANSI standard or that support a block size smaller than 2048 bytes. For example, you must specify /BLOCKSIZE=512 to mount an RT–11 volume.

If you want to write files on a magnetic tape volume, use the /BLOCKSIZE qualifier to set the block size to other than 2048. If you want only to read files from a magnetic tape volume, you do not have to use the /BLOCKSIZE qualifier, since VMS will automatically handle the block size given by the second file header label.

The minimum blocksize for ANSI standard magnetic tapes is 18, while the maximum is 2048. The minimum blocksize for VMS magnetic tapes is 14, while the maximum is 65,532.

/FOREIGN

The /FOREIGN qualifier should be used when a magnetic tape volume is not in standard ANSI or ISO format or when a disk volume is not in Files–11 format.

/OVERRIDE

The /OVERRIDE=(option[,...]) qualifier inhibits one or more of the access checks performed by MOUNT and the magnetic tape file system. The options are as follows:

- ACCESSIBILITY — If the installation allows, this option will override any character in the accessibility field of the volume and file header labels. The necessity of this qualifier is defined by the installation. That is, each installation has the option of specifying a routine that the magnetic tape file system will use to process this field. By default, the VMS operating system provides a routine that checks this field in the following manner:
  - If the magnetic tape was created on a version of the VMS operating system that conforms to Version 3 of ANSI, then this option must be used to override any character other than an ASCII space.
  - If a VMS protection is specified and the magnetic tape conforms to an ANSI standard that is later than Version 3, then this option must be used to override any character other than an ASCII 1.

  For more information on accessibility-based protection for VMS ANSI-labeled magnetic tapes, see Chapter 2.
- EXPIRATION — Overrides the expiration dates of a volume and its files. Use this qualifier when the expiration date (in the first file-header label) of any file that you want to overwrite has not been reached.
- IDENTIFICATION — Overrides the volume identifier in the volume label. Use this qualifier to mount a volume for which you do not specify the volume identifier. Only the volume identifier field will be overridden. Volume protection, if any, is preserved.
- OWNER_IDENTIFIER — Overrides the processing of the owner identifier field. You use this option when you need to interchange protected magnetic tapes between VMS and other DIGITAL operating systems.
- SETID — Prevents MOUNT from checking the file-set identifier in the first file-header label of the first file on a continuation volume. Use this qualifier only for ANSI-labeled volumes on which the file-set identifier of the first file on a continuation volume differs from the file-set identifier of the first file of the first volume that was mounted.

/OWNER_UIC=uic

The /OWNER_UIC=uic qualifier overrides the UIC written in the second volume label and assigns the UIC you specify while the volume is mounted. For magnetic tape volume sets in which a continuation volume is written, the UIC specified at mount time is written to the volume only if the /PROTECTION qualifier was specified either at mount time or when the volume was initialized. This does not change the protection on any volumes already created.

You can specify the UIC variable in the format:

[g,m]

g is either an octal number in the range 0 through 37776 that denotes the group number or an alphanumeric value (consisting of 1 through 31 characters) that describes the group.

m is either an octal number in the range 0 through 177776 that denotes the member number or an alphanumeric value (consisting of 1 through 31 characters) that describes the member.

Either square ([ ]) or angle (<>) brackets are required in the UIC specification. For more details on UIC-based protection, see Chapter 2.

/CACHE=TAPE_DATA

The /CACHE qualifier with the TAPE_DATA option enables the write cache for a tape device if the tape controller supports one. /NOCACHE is the default for mounting tape devices. You must specify TAPE_DATA to enable the write cache. If the tape controller does not support a write cache, the option is ignored. This lets you specify /CACHE=TAPE_DATA in command procedures that work with a variety of devices and controllers.

This option enables a form of controller-based, write-back caching. The write-back caching feature significantly improves the overall performance of streaming tape drives. Under some rare failure conditions, however, some written data can be lost in the cache of the controller. If a failure occurs, the magnetic tape being written becomes seriously flawed or unreadable, and you must repeat the entire process used to write to the tape. When a failure occurs, you are always notified with an error message.

/PROTECTION=code

The /PROTECTION=code qualifier overrides the VMS protection written in the second volume label and assigns the protection code you specify to the volume while it is mounted. For magnetic tape volume sets in which a continuation volume is written, the protection code you specify will be written to the continuation volume. By default, your process UIC also will be written to the continuation volume unless you explicitly specify an alternate UIC with the /OWNER_UIC qualifier described above.

Valid protection codes include READ and WRITE access for GROUP and WORLD users; EXECUTE and DELETE access are not applicable to magnetic tape volumes. SYSTEM users and the volume owner always have READ and WRITE access to magnetic tape volumes regardless of the protection code that you specify. Section 2.2.2 describes access and protection codes.

/HDR3

The /NOHDR3 qualifier controls whether special VMS header labels are written to a volume. Privilege is not required for this qualifier. The default is /HDR3, which allows VMS header labels to be written to a volume. When the /NOHDR3 qualifier is used, long VMS file names are truncated to 17 characters. Use the /NOHDR3 qualifier when writing to volumes that will be read by a system other than VMS, such as the RT–11 system, which does not process all file-header labels correctly.

/RECORDSIZE=n

The /RECORDSIZE=n qualifier specifies the number of bytes in each record. This qualifier does not require privilege. Use this qualifier when you mount volumes without the second file-header label (such as RT–11 volumes), or when you mount volumes with the /FOREIGN qualifier, to provide RMS with the size of fixed-length records or the maximum size of variable-length records.

The record size must be less than or equal to the specified or default block size. Refer to the /BLOCKSIZE qualifier (described previously) for details. The VMS operating system does not write records smaller than 14 bytes on output. However, the VMS Convert Utility, described in the *VMS Convert and Convert/Reclaim Utility Manual*, allows you to pad and extend the size of records up to and greater than the 14-byte minimum.

Two other qualifiers that are important for mounting magnetic tape volumes are /AUTOMATIC and /INITIALIZE. These qualifiers are described in the following sections.

3.4.4 Mounting a Magnetic Tape Volume Set

When mounting a magnetic tape volume set, begin by following the procedures described in Section 3.4.3 for mounting a single magnetic tape volume. The number of volume identifiers need not equal the number of device names specified. Thus, when you mount a magnetic tape volume set, you can specify more volume identifiers than device names or more device names than volumes.

The number of devices you specify directly affects the action taken by the magnetic tape file system when processing continuation volumes in a volume set. For example, when the number of devices is greater than the number of volumes, the magnetic tape files system requests a continuation volume to be mounted on the first drive from the list that does not have a volume mounted.

The next two sections describe how to create and mount a magnetic tape volume set. The manner in which continuation volumes are handled by the magnetic tape file system is also described.

3.4.4.1 Creating a Magnetic Tape Volume Set

If you do not create a volume set explicitly, the VMS system creates one when necessary. If you have not mounted a volume set and a continuation volume is required, the magnetic tape file system requests that a continuation volume be mounted and implicitly creates a volume set. For example, if the magnetic tape file system encounters an EOT mark while writing a volume, it sends a message to the operator console requesting that another volume be mounted.

After you mount the next volume, the magnetic tape file system writes the volume and header labels and then reissues the pending write requests to the continuation volume. The file-set identifier in the first file-header label of all files written to the continuation volume is the file-set identifier of the first file on the first volume. The file-set identifier for VMS volume sets is always that of the first file of the first volume that was mounted in the set.

To explicitly create a volume set with three volumes, follow this procedure:

1 Allocate a drive on which you will load each volume by entering the following ALLOCATE commands:

```
$ ALLOCATE MTA0:
%DCL-I-ALLOC, _MARS$MTA0: allocated

$ ALLOCATE MTA1:
%DCL-I-ALLOC, _MARS$MTA1: allocated

$ ALLOCATE MTA2:
%DCL-I-ALLOC, _MARS$MTA2: allocated
```

2 Initialize the volumes. You should specify the density and the access protection in addition to the device name and the volume identifier in the INITIALIZE commands, as in the following command lines:

```
$ INITIALIZE/DENSITY=1600/PROTECTION=(G:RW) MTA0: TAPE1
$ INITIALIZE/DENSITY=1600/PROTECTION=(G:RW) MTA1: TAPE2
$ INITIALIZE/DENSITY=1600/PROTECTION=(G:RW) MTA2: TAPE3
```

3 Mount the volumes by entering the following MOUNT command. You should include the device name and volume identifier. Specifying a logical name for the volume set is optional.

```
$ MOUNT MTA0:,MTA1:,MTA2: TAPE1,TAPE2,TAPE3 TEST
%MOUNT-I-MOUNTED, TAPE1 mounted on _MTA0:
%MOUNT-I-MOUNTED, TAPE2 mounted on _MTA1:
%MOUNT-I-MOUNTED, TAPE3 mounted on _MTA2:
```

The system not only confirms which volumes have been mounted but also indicates on which drive each volume has been mounted.

The system mounts and verifies only the volumes that are physically loaded on the drives at mount time. However, the volume identifiers of additional volumes that you specify are not verified until the volumes are accessed.

You can check the densities, volume labels, UICs, and relative volume numbers of the volumes that are mounted on drives. To do so, specify the SHOW DEVICES/FULL command. If you specify a generic device code for the magnetic tape drives, such as MT, information for all the drives of that type configured in the system is displayed. To display information for a volume mounted on a specific drive, specify the physical device code, consisting of the generic device code, the controller designation, and the unit

number followed by a colon. For more information on the SHOW DEVICES command, including examples of displays returned by the SHOW DEVICE /FULL command, see Chapter 4 and the *VMS DCL Dictionary*.

3.4.4.2 Mounting Continuation Volumes in a Volume Set

When mounting a magnetic tape volume set, follow the general procedures described in the previous section for creating a magnetic tape volume set. Once the volume set has been created, however, there is no need to initialize the volumes in the set when you mount the volume set.

You need not allocate a drive for each volume in the volume set. The magnetic tape file system requests that volumes be switched to appropriate drives when continuation volumes are required.

The VMS operating system stores, but cannot verify, the volume identifiers of volumes you specify but do not physically mount on drives at mount time. VMS later verifies the volume identifiers of such volumes when the volumes are accessed.

The VMS operating system supports the continuous processing of mounted volumes in a magnetic tape volume set through automatic volume switching. To do this, the magnetic tape file system uses automatic volume recognition (AVR) and automatic volume labeling (AVL).

To take advantage of this automatic volume switching capability, you must have more than one magnetic tape drive allocated to your volume set. If you have two or more magnetic tape drives allocated to a volume set, the magnetic tape file system switches volumes for you automatically by sequentially selecting the next magnetic tape drive allocated to the volume set. The magnetic tape file system expects the next volume in the volume set to be loaded on that drive.

If the file system is writing to the volume set, it creates a label for the magnetic tape and initializes the magnetic tape with that label and with the protection characteristics set for the first of the volume set. If the magnetic tape file system is reading the volume set, it generates the label and tries to mount the next magnetic tape with that label. If the drive has no magnetic tape loaded on it, or the wrong magnetic tape, the magnetic tape file system sends a message to the operator console notifying the operator to either mount a magnetic tape or mount the correct magnetic tape.

Before processing continuation volumes, the magnetic tape file system processes the protection on that volume (as described in Section 2.1.3). If the magnetic tape file system determines that the user does not have access to the volume, then a message is sent to the operator to take some action.

The label generated fills the six-character volume identifier field. The first four characters of the field contain the first four characters of the label specified for the previous volume in the volume set. (If the label is less than four characters, the volume identifier field is padded with underscores.) The fifth and sixth characters contain the relative volume number for that reel in the volume set. Note that this allows VMS to generate only 99 unique labels for a given volume set.

With automatic volume switching enabled, the operator can load a magnetic tape on the next drive allocated to the magnetic tape volume set anytime before the volume being processed reaches the EOT mark. The magnetic tape file system mounts and initializes (if INITIALIZE was specified originally) the next magnetic tape in the volume set and then notifies the operator that the switch has occurred.

In the following example, the volume with the identifier TAPE is mounted on MTA0:

```
$ MOUNT MTA0:,MTA1:,MTA2:  TAPE
```

Continuation volumes for this set should be loaded on the magnetic tape drives in the following order: MTA1, MTA2, MTA0, MTA1, MTA2, and so forth.

To explicitly override automatic volume switching, specify the /NOAUTOMATIC qualifier when mounting a magnetic tape volume. The default is /AUTOMATIC. If you allocate only one drive to the magnetic tape volume set, you implicitly disable automatic volume switching.

To ensure that any volume added to the magnetic tape volume set will be initialized prior to being written to, mount the volume with the /INITIALIZE=CONTINUATION qualifier. The default is /NOINITIALIZE.

The next example demonstrates the use of the /INITIALIZE=CONTINUATION qualifier for mounting volume sets. It also shows how volume identifiers are generated for continuation volumes.

```
$ INITIALIZE MTA0: MAIN
$ MOUNT/OVERRIDE=IDENTIFIER/INITIALIZE=CONTINUATION MTA0:,MTA1:
```

The volume labeled MAIN is mounted on MTA0. The second volume in the set receives the volume identifier MAIN02 and is mounted on MTA1. The third volume in the set receives the volume identifier MAIN03 and is mounted on MTA0. In the following example, the first volume in the set is labeled SUN and is mounted on MTA0. The second volume receives the identifier SUN_02 and is mounted on MTA1. The third volume receives the identifier SUN_03 and is mounted on MTA0:

```
$ MOUNT MTA0:,MTA1: SUN
```

The next example illustrates a continuation volume with two volume identifiers.

```
$ MOUNT MTA0:,MTA1: SUN,MOON
```

In this case, SUN and MOON are mounted on MTA0 and MTA1 respectively. If a third volume is added to the set, it will be given the identifier MOON03 and be mounted on MTA0.

3.5 Dismounting a Volume

When you have finished processing the files or data on your disk or magnetic tape volume, you can use the DISMOUNT command to explicitly dismount a single volume or an entire volume set.

If you explicitly dismount a single volume in a volume set, VMS dismounts all the volumes in the set. For disk volume sets, however, it is possible to explicitly dismount a single volume in the volume set without dismounting the entire set. To do this, you must use the /UNIT qualifier.

When you enter the DCL command DISMOUNT, the volume is automatically unloaded from the drive. You can override this automatic unloading of your volume by specifying the /NOUNLOAD qualifier with the DISMOUNT command.

Even when you specify the /NOUNLOAD qualifier with the DISMOUNT command, your volume is still logically dismounted from the drive; however, the volume remains physically loaded on the drive. If you use the /NOUNLOAD qualifier to dismount a magnetic tape volume, the volume remains loaded on the magnetic tape drive and the magnetic tape reel is rewound to the BOT mark.

If you plan to remount or reinitialize a volume you are dismounting, you can save time and eliminate unnecessary handling of that volume by using the /NOUNLOAD qualifier with the DISMOUNT command.

The following examples show how to use the DISMOUNT command. The first example employs the /NOUNLOAD qualifier.

```
$ DISMOUNT/NOUNLOAD MTA1:
$
```

In this example, the magnetic tape volume is logically dismounted and remains loaded on drive MTA1. Also, the magnetic tape reel is rewound to the BOT mark. The VMS operating system returns you to DCL level.

The DISMOUNT command is also used to dismount foreign volumes. The following command dismounts a volume that had been mounted with the /FOREIGN qualifier on DBA0:

```
$ DISMOUNT DBA0:
$
```

In this example, the volume that had been mounted with the /FOREIGN qualifier on DBA0 is dismounted and automatically unloaded. The VMS system returns you to DCL level.

As mentioned previously, use the DISMOUNT command to dismount an entire volume set. If you explicitly dismount any volume in a disk or magnetic tape volume set, the entire volume set is dismounted. For example, if you had a volume set that consisted of DBA3 and DBA4 and you entered the following command, the entire volume set would be dismounted:

```
$ DISMOUNT DBA3:
```

You should always explicitly dismount a volume or volume set with the DISMOUNT command, or with a command procedure containing that command, before physically unloading that volume.

A volume is dismounted and unloaded automatically if you log out of the job from which you had mounted the volume. If the system fails, however, the drive is not automatically dismounted.

Note that data corruption can occur if a volume has not been explicitly dismounted and the system fails. For magnetic tape volumes, data corruption can occur if you unload a volume that contains an open file for which file-trailer labels have not been written. When you remount the volume and attempt to access the file without file-trailer labels, you receive the following error message:

```
%MTAACP-magnetic tape position lost
```

You will be able to access all the files (on that magnetic tape volume) preceding the file whose file-trailer labels had not been written. However, you will not be able to access the file without file-trailer labels.

Note that the dismount of a volume is done by the file system and is not complete until all the open files on the volume have been closed. Thus, a substantial amount of time can pass between the time you enter the DISMOUNT command and the completion of the dismount. Always wait for the drive to unload before you remove the volume. (You can verify that the dismount is complete by entering the DCL command SHOW DEVICES.)

If the device you are dismounting was allocated with an ALLOCATE command, it remains allocated after it is dismounted with the DISMOUNT command. If the device was implicitly allocated by the MOUNT command, the DISMOUNT command deallocates it.

For more information on the DISMOUNT command, see the *VMS DCL Dictionary*.

3.6 Deallocating Drives

The process of allocation reserves a device for exclusive use by your process. The device remains allocated to your process until you explicitly deallocate it or until you log out from your process. Once you have allocated the device, other users cannot access that device until you explicitly deallocate it or log out.

Use the DCL command DEALLOCATE to explicitly deallocate a disk drive or magnetic tape drive that has been allocated to your process. A complement to the ALLOCATE command, the DEALLOCATE command logically disconnects a drive from your process and returns it to the pool of devices.

The following example shows how to explicitly deallocate a magnetic tape drive or a disk drive:

```
$ DEALLOCATE MTA1:
$
```

In this example, the DEALLOCATE command logically disconnects magnetic tape drive MTA1 from your process. The VMS operating system returns you to DCL level.

Since logging out of a process from which drives have been allocated automatically deallocates all explicitly and implicitly allocated drives, you do not have to explicitly deallocate a disk or magnetic tape drive that has been allocated to your process. However, it is a good practice to use the DEALLOCATE command (or a command procedure containing this command) to explicitly deallocate all the drives you allocated with the ALLOCATE command.

3.7 Using Command Procedures to Set Up Volumes

Since private disk and magnetic tape volumes frequently must be set up before you can perform operations on them, you may want to design command procedures to facilitate the set-up procedures. This section contains examples of command procedures that can be used to set up disk and magnetic tape volumes for routine operations.

You can tailor command procedures to meet the needs of your own set-up tasks. The command procedure examples in this section, although general in nature, can serve as guiding strategies for you.

The next two sections contain examples of command procedures designed to set up disk and magnetic tape volumes for routine processing.

3.7.1 Designing Command Procedures to Set Up Disk Volumes

The following command procedure is designed to allocate, initialize, and mount a disk volume. You can use a text editor, such as EDT or EVE, to create a file to contain your command procedure. Assume that a file named SETUP.COM has been created and that it contains a very basic command procedure, which, when executed, allocates and mounts a disk. Construct this command procedure by entering the following text:

```
$ ! Place a disk in the drive
$ IF P1 .EQS. "" THEN INQUIRE P1 "enter device name"
$ IF P2 .EQS. "" THEN INQUIRE P2 "enter volume label"
$ IF P3 .EQS. "" THEN INQUIRE P3 "enter logical name"
$ ALLOCATE 'P1'
$ MOUNT 'P1' 'P2' 'P3'
```

This command procedure, although very simple, accomplishes the task of allocating and mounting a disk each time it is executed. It is designed to prompt you for the device name, volume label, and logical name of the disk device that you want to allocate and mount. By assigning logical names to your disks, you can use this command procedure to allocate and mount devices over and over again.

You can take further advantage of the power of a command procedure by including a few additional tasks as well. For example, you could design the SETUP.COM command procedure to deallocate and dismount the disk. The command procedure example used to set up a magnetic tape (described in the next section) takes advantage of some of these options.

To execute the SETUP.COM command procedure, enter the following command:

```
@SETUP
```

You can also write command procedures to mount a volume from a batch job. By using logical names to refer to devices and files, you can use the same command procedures without modification each time you want to access a volume.

For example, if you use the same RK07 disk pack to back up your files on a weekly basis, you can submit as a batch job a command procedure such as the following:

```
$ MOUNT  DM:  BACK_UP_GMB  RK
$ BACKUP/REPLACE  *.*  RK:*.*
$ DIRECTORY/FULL/OUTPUT=BACKUP.LOG  RK:
$ DISMOUNT RK:
```

In this command procedure, the MOUNT command finds and allocates a device of the type DM and creates a logical name RK. The MOUNT command substitutes the equivalence name in the message displayed at the operator console.

When the MOUNT command notifies the operator to mount the correct volume, the job waits until the operator responds. When the operation is completed, the DISMOUNT command dismounts the device RK.

3.7.2 Designing Command Procedures to Set Up Magnetic Tape Volumes

The command procedure below, which is more sophisticated and detailed than the previous example, is designed to set up a magnetic tape for processing. The ALLOCATE and MOUNT/FOREIGN commands are included in this command procedure. Using a text editor, construct the command procedure in the following way:

```
$ ! First mount the tape on the drive
$ ON CONTROL_Y THEN GOTO EXIT
$ ON ERROR THEN GOTO EXIT
$ WRITE SYS$OUTPUT "Welcome to FETCH."
$ WRITE SYS$OUTPUT " "

$ L1:  INQUIRE/NOPUNC PHYS "Have you placed the volume in the drive? "
$ IF .NOT. PHYS THEN GOTO L1
$ INQUIRE/NOPUNC DRIVE "Which drive is the volume mounted on? "
$ DRIVE = DRIVE - ":"
$ ALLOCATE 'DRIVE'
$ MOUNT/FOREIGN 'DRIVE'
$ ON ERROR THEN GOTO COMMAND_LOOP
$ !
$ COMMAND_LOOP:  INQUIRE/NOPUNC OPTION "FETCH> "

$ IF OPTION .EQS. "DIR" THEN GOTO DIR
$ IF OPTION .EQS. "EXIT" THEN GOTO EXIT
$ IF OPTION .EQS. "FETCH" THEN GOTO FETCH
$ IF OPTION .EQS. "HELP" THEN GOTO HELP
$ IF OPTION .EQS. "LIST" THEN GOTO LIST
$ GOTO COMMAND_LOOP
$ !
$ DIR:  INQUIRE SPEC "Filespec"
$ DIR 'SPEC'
$ GOTO COMMAND_LOOP
$ HELP:
$ WRITE SYS$OUTPUT "Enter any of the following commands at the prompt:"
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT "DIR         (To search for a file)"
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT "EXIT        (To exit this program)"
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT "FETCH       (To perform a BACKUP RESTORE operation)"
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT "HELP        (To read this text)"
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT "LIST        (To perform a BACKUP LIST operation)"
$ GOTO COMMAND_LOOP
$ !
$ FETCH:  INQUIRE FILE "Filespec"
$ INQUIRE SAVESET "Save set name"
$ LINE := BACKUP/LOG 'DRIVE':'SAVESET'/SELECT='FILE'
$ INQUIRE EXCLUDE "Enter any filespecs you want excluded"
$ IF EXCLUDE .EQS. "" THEN GOTO L2
$ LINE := 'LINE'/EXCLUDE=('EXCLUDE')
```

```
$ !
$ L2:  INQUIRE/NOPUNC TO "Where do you want the file(s)? (RET for current directory)"
$ IF TO .EQS. "" THEN GOTO REPLACE
$ LINE := 'LINE' 'TO'
$ GOTO L3
$ REPLACE:  LINE := 'LINE' []
$ !
$ L3:  INQUIRE/NOPUNC NEW "Create a new version if file already exists? "
$ IF .NOT. NEW THEN GOTO NOT
$ LINE := 'LINE'/NEW_VERSION
$ !
$ NOT:  LINE := 'LINE'/OWNER_UIC=ORIGINAL
$ LINE
$ GOTO COMMAND_LOOP
$ !
$ LIST:  INQUIRE SPEC "Filespec"
$ INQUIRE SAVESET "Save set name"
$ INQUIRE/NOPUNC OUTPUT "What do you want to call the list file? (RET for SYS$OUTPUT ")
$ IF OUTPUT .EQS. "" THEN GOTO NOOUT
$ LINE := BACKUP/LIST='OUTPUT' 'DRIVE':'SAVESET'/SELECT=('SPEC')
$ GOTO L4
$ NOOUT:  LINE := BACKUP/LIST 'DRIVE':'SAVESET'/SELECT=('SPEC')
$ !
$ L4:  INQUIRE EXCLUDE "Enter any filespecs you want excluded"
$ IF EXCLUDE .EQS. "" THEN GOT L5
$ LINE := 'LINE'/EXCLUDE=('EXCLUDE')
$ !
$ L5:  LINE
$ GOTO COMMAND_LOOP
$ !
$ EXIT:
$ DISMOUNT 'DRIVE'
$ DEALLOCATE 'DRIVE'
```

Assume this command procedure is contained in a file named FETCH.COM; you would execute this command procedure by entering the following:

```
@FETCH
```

This command procedure is more complex than the two previous ones, which were both used to set up a disk. As with the former command procedures, the procedure contained in FETCH.COM also accomplishes the basic task of allocating the device and mounting the volume.

In addition to its allocating and mounting functions, the command procedure contained in FETCH.COM is designed to prompt you for input. For example, it specifically asks you if the magnetic tape is on the drive. Also, note that this command procedure is designed to do a BACKUP restore operation. It prompts you for specific options on the restore operation.

Finally, this command procedure explicitly dismounts your magnetic tape volume and deallocates the drive after your task has completed.

4 Manipulating Files

You can manipulate disk and magnetic tape files by using the DIGITAL Command Language (DCL). In particular, you can use DCL commands to perform the following tasks:

- Retrieve disk and magnetic tape file information
- Modify disk and magnetic tape file characteristics
- Access files residing on disk and magnetic tape volumes

DCL commands enable you to manipulate files in the following ways:

- SHOW commands retrieve disk and magnetic tape file information, such as device and protection characteristics, and display this information on your terminal.
- SET commands modify disk and magnetic tape file characteristics, such as protection or UIC information.
- The DCL command language can be used to access disk and magnetic tape files for read or write operations.

In addition to manipulating files through DCL, you can write user programs to assist you in routine file-manipulation tasks. You can write these programs in either VAX MACRO or in one of the higher-level languages supported by the VMS operating system. If you want to manipulate individual records within files — that is, to access files at the record level — you should write programs that include RMS facilities. Examples of the RMS facilities used to manipulate files at the record level are included in the *VMS Record Management Services Manual*.

You can use DCL to manipulate disk and magnetic tape files at the file level. Note the following restrictions on the use of DCL commands for manipulating files on disk and magnetic tape volumes:

- The SUBMIT command cannot access files on allocated devices. You can submit files on private disk volumes if you mount the volume as a shareable volume. To submit a file on a magnetic tape volume, you first must copy the file to a shared disk volume and then submit it from the disk.
- You can print a file from a privately owned volume. Note, however, that the volume containing the file you wish to print must remain mounted until after the file has completed printing. If you do not want to wait until the printing completes, you can copy the file directly to the line printer, using the DCL command COPY as in the following example:

  ```
  $ COPY MYPHILE.DAT LPAO:
  ```

- Most DCL commands require file-structured devices. For a list of those commands that do not require file-structured devices, see the *VMS DCL Dictionary*.

- You can execute a command procedure that resides on a magnetic tape volume as long as the procedure does not invoke other procedures and does not issue any GOTO commands that refer to labels in the procedure preceding the GOTO command. In this case, it would be better to copy the command procedure from the magnetic tape volume to a local disk from which you can then invoke the command procedure.

Note that you cannot use DCL commands to read or write files that are not in the standard formats supported by VMS (these formats are described in greater detail in the *Guide to Using VMS Command Procedures*).

4.1 Using DCL to Retrieve File Information

The DCL command language provides commands that enable you to retrieve information about disk and magnetic tape files, volumes, and devices. You can use the following DCL commands to retrieve such information:

- DIRECTORY
- SHOW DEVICES
- SHOW MAGTAPE
- SHOW ACL
- SHOW PROTECTION
- SHOW QUOTA

See the *VMS DCL Dictionary* for a complete list of the command qualifiers and parameters applicable to each of these DCL commands.

4.1.1 Retrieving Directory Information

Use the DCL command DIRECTORY to retrieve information about a file or a group of files residing on a disk or magnetic tape volume.

You can use the DIRECTORY command to list the names of all of the files in a particular directory, or you can use it in conjunction with a file specification to list the names of specific files in a given directory. When you include certain command qualifiers along with the DIRECTORY command, you can retrieve information in addition to the names of the files. See the *VMS DCL Dictionary* for a list of qualifiers that can be used with the DIRECTORY command.

The following examples illustrate three cases of retrieving information from the [MALCOLM] directory, which resides on a disk with the logical name DISK$DOCUMENT.

Examples

1

```
$ DIRECTORY AVERAGE.*

Directory DISK$DOCUMENT:[MALCOLM]

AVERAGE.EXE;6      AVERAGE.FOR;6      AVERAGE.LIS;4      AVERAGE.OBJ;12

Total of 4 files.
```

2

```
$ DIRECTORY/SIZE=USED/DATE=CREATED/VERSIONS=1/PROTECTION  AVERAGE

Directory DISK$DOCUMENT:[MALCOLM]

AVERAGE.EXE;6        6        10-APR-1988 15:43 (RWED,RWED,RWED,RE)
AVERAGE.FOR;6        2         2-APR-1988 10:29 (RWED,RWED,RWED,RE)
AVERAGE.LIS;4        5         9-APR-1988 16:27 (RWED,RWED,RWED,RE)
AVERAGE.OBJ;6        2         9-APR-1988 16:27 (RWED,RWED,RWED,RE)

Total of 4 files, 15 blocks.
```

3

```
$ DIRECTORY/FULL/VERSIONS=1    [MALCOLM...]AVERAGE.EXE

Directory DISK$DOCUMENT:[MALCOLM]

AVERAGE.EXE;6                 File ID:  (4098,149,0)
Size:           36/36         Owner:    [DOCUMENTATION,MALCOLM]
Created:  27-JUN-1988 12:22:26.30
Revised:  27-JUN-1988 12:22:51.35 (2)
Expires:   <None specified>
Backup:    3-JUL-1988 22:03.09
File organization:  Sequential
File attributes:    Allocation: 36, Extend: 36, Global buffer count: 0
                    No version limit
Record format:      Variable length, maximum 255 bytes
Record attributes:  Carriage return carriage control
Journaling Enabled : None
File protection:    System:RWED, Owner:RWED, Group:RE, World:
Access Cntrl List:  None

Total of 1 file, 36/36 blocks.

Directory DISK$DOCUMENT:[MALCOLM.TEST]

AVERAGE.EXE;1                 File ID:  (7714,29,0)
Size:           36/36         Owner:    [DOCUMENTATION,MALCOLM]
Created:  15-APR-1988 10:12
Revised:  15-APR-1988 10:12 (1)
Expires:   <None specified>
Backup:   15-APR-1988 22:41
File organization:  Sequential
File attributes:    Allocation: 36, Extend: 36, Global buffer count: 0
                    No version limit
Record format:      Variable length, maximum 255 bytes
Record attributes:  Carriage return carriage control
Journaling Enabled : None
File protection:    System:RWED, Owner:RWED, Group:RE, World:
Access Cntrl List:  None

Total of 1 file, 36/36 blocks.

Grand total of 2 directories, 2 files, 72/72 blocks.
```

Directory structures do not apply to magnetic tape volumes. However, you can use the DIRECTORY command to search for files on magnetic tape volumes. This use of the DIRECTORY command is also discussed in Section 4.3.2, which describes how to access magnetic tape files for read and write operations.

To find the names of all files on a magnetic tape volume mounted on MTA2, enter the following command:

```
$ DIRECTORY MTA2:
```

This directory command lists the file names and file types of all files on the magnetic tape.

As in the case of the disk examples included above, you can use wildcard characters in directory specifications for magnetic tapes, as in the following command:

```
$ DIRECTORY MFA0:*.*;*
```

In response to this command, the VMS operating system searches the entire volume set and returns both ANSI and VMS file names. (The difference between these two types of magnetic tape files is described in Section 4.3.2).

4.1.2 Retrieving Device Information

Use the DCL command SHOW DEVICES to retrieve information about the availability of devices on your system.

When you enter the SHOW DEVICES command without specifying a device or using a qualifier, information about all devices on the system is displayed. If you specify a device name, SHOW DEVICES displays information about that device. If you use certain qualifiers with SHOW DEVICES, information is displayed about those devices that currently have volumes mounted or that have been allocated to processes. See the *VMS DCL Dictionary* for a list of qualifiers that can be used with the SHOW DEVICES command.

The device name displayed by the system uses the format *ddcu*, where *dd* is the device code, *c* is the control, and *u* is the unit. If the system is part of a VAXcluster environment, the device name includes the node name in the format *node$ddcu*; where, *node* refers to the node name of the system that the device resides on, *dd* refers to the device type, *c* refers to the controller designation, and *u* refers to the unit number. See the *VMS VAXcluster Manual* for a complete description of the device name format on clusters.

The following examples illustrate three instances of how the SHOW DEVICES command can be used.

Examples

1

```
$ SHOW DEVICES

Device                  Device      Error    Volume          Free   Trans  Mnt
 Name                   Status      Count     Label          Blocks Count  Cnt
DBA0:                   Mounted         0  VMS             47088    115    1
DBA1:                   Mounted         0  USERPACK1       45216      2    1
DBA2:                   Mounted         3  DOCUMENT         8068     20    1
DBA5:                   Mounted         0  MASTERP         28668      1    1
DBA6:                   Online          0
DBA7:                   Mounted         0  PROJECT        110547      1    1
DMA0:                   Online          0
DLA0:                   Online          0
DYA0:                   Online          0
DYA1:                   Online          0
DRA3:                   Mounted         0  RES26APR        29317      1    1
MFA0:                   Online          8
MFA1:                   Online          1
MTA0:                   Mounted         9  BACKUP            453      1    1
MTA1:                   Online          0
```

The SHOW DEVICES command displays the following information for each device on the system:

- Device name.
- Device status and characteristics. (Status indicates whether the device is on line; characteristics indicate whether the device is allocated, spooled, or has a volume mounted on it and if the volume is mounted foreign.)
- Error count.
- Volume labels.
- Number of free blocks on the volume (disk only).
- Transaction count.
- Number of mount requests issued for the volume.

2

```
$ SHOW DEVICES/FULL DMA0

Disk $1$DMA0: (NODE1), device type RK07, is online, allocated,
     served to the cluster, error logging enabled.

    Error count                    2    Operations completed                4527
    Owner process              "SMITH"  Owner UIC                          [0,0]
    Owner process ID          24400133  Dev Prot    S:RWED,O:RWED,G:RWED,W:RWED
    Reference count                1    Default buffer size                  512
    Allocation class               1
```

This SHOW DEVICES command requests a full listing of the status of the RK07 device DMA0. The device is located on NODE1 in a VAXcluster.

3

```
$ SHOW DEVICES/FULL NODE2$

Disk $1$DBA0: (NODE2), device type RP05, is online, mounted, file-oriented
    device, shareable, served to cluster via MSCP Server, error logging is
    enabled.

    Error count                    0    Operations completed                120
    Owner process                 ""    Owner UIC                         [303,2]
    Owner process ID        00000000    Dev Prot    S:RWED,O:RWED,G:RWED,W:RWED
    Reference count                1    Default buffer size                 512
    Total blocks              171798    Sectors per track                    22
    Total cylinders              411    Tracks per cylinder                  19
    Allocation class               1

    Volume label          "HIGHNOON"    Relative volume number               0
    Cluster size                   3    Transaction count                    1
    Free blocks                94425    Maximum files allowed            21474
    Extend quantity                5    Mount count                          8
    Mount status              System    Cache name        "_$255$DUA8:XQPCACHE"
    Extent cache size             64    Maximum blocks in extent cache    9442
    File ID cache size            64    Blocks currently in extent cache     0
    Quota cache size               0    Maximum buffers in FCP cache       421

  Volume status:  subject to mount verification, file high-water marking, write-
      through caching enabled.
  Volume is also mounted on NODE1, NODE4, NODE3.

Disk NODE2$DBC1:, device type RP06, is online, error logging enabled.

    Error count                    0    Operations completed                  0
    Owner process                 ""    Owner UIC                         [0,0]
    Owner process ID        00000000    Dev Prot    S:RWED,O:RWED,G:RWED,W:RWED
    Reference count                0    Default buffer size                 512
    Host name                "NODE2"    Host type, available          V780, yes

Disk NODE2$DMA0:, device type RK07, is online, error logging enabled.

    Error count                    0    Operations completed                  0
    Owner process                 ""    Owner UIC                         [0,0]
    Owner process ID        00000000    Dev Prot    S:RWED,O:RWED,G:RWED,W:RWED
    Reference count                0    Default buffer size                 512
    Host name                "NODE2"    Host type, available          V780, yes
    .
    .
    .
```

The command line SHOW DEVICES/FULL NODE2$ produces a full display of information about each device on NODE2 on the VAXcluster system. Information is shown here only for the first three devices: a mounted device and two that are not mounted.

4.1.3 Retrieving Magnetic Tape Device Information

You can use the DCL command SHOW MAGTAPE to display the current characteristics and status of a specified magnetic tape device.

You can enter the SHOW DEVICES command to find available magnetic drives on your system. The SHOW MAGTAPE or SHOW DEVICE /FULL commands enable you to retrieve additional information about the characteristics of a particular magnetic tape device.

When you enter the SHOW MAGTAPE command at your terminal, you receive the following prompt:

```
_Device:
```

You must then specify the name of the magnetic tape device for which you want to display the characteristics and status.

The following example illustrates how the SHOW MAGTAPE command is used to retrieve information about MTA0:

Example

```
$ SHOW MAGTAPE MTA0:
MTA0:,          device type TU77, is online, error logging is enabled

Error count              0    Operations completed                0
Owner process          " "    Owner UIC                       [0,0]
Owner process ID 0000000    Dev Prot S:RWED, O:RWED, G:RWED, W:RWED
Reference Count          0    Default buffer size              2048
Density                800    Format                      Normal-11
Volume Status: no-unload on dismount, odd parity)
```

This SHOW MAGTAPE command displays the characteristics of the device MTA0. Among other characteristics, it displays the device type, density, and format.

4.1.4 Retrieving Disk File Protection Information

The DCL command SHOW PROTECTION displays the current process default protection. This protection is applied to files created during your terminal session or to batch jobs, where defaults from directories or previously existing versions are not available.

You can change the default protection at any time with the SET PROTECTION command. (The SET ACL and SET PROTECTION commands are discussed in Section 4.3, which describes how to modify file characteristics.)

This section is not applicable to magnetic tapes. Although you can use the SHOW PROTECTION or SET PROTECTION commands to show or set the default protection of magnetic tapes, the protection is not written to the magnetic tape volume unless you specify the /PROTECTION= qualifier with the INITIALIZE command when you are initializing the magnetic tape volume. See the description of initializing magnetic tape volumes in Chapter 3.

The next example illustrates how the SHOW PROTECTION command can be used in conjunction with the SET PROTECTION command to display and modify the protection characteristics of a disk file.

Example

```
$ SHOW PROTECTION
SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=NO ACCESS
$ SET PROTECTION=(GROUP:RWED,WORLD:RE)/DEFAULT
$ SHOW PROTECTION
SYSTEM=RWED, OWNER=RWED, GROUP=RWED, WORLD=RE
```

This SHOW PROTECTION command requests a display of the current protection defaults. The SET PROTECTION/DEFAULT command is then used to change the file access allowed to other users in the same group as well as to miscellaneous system users.

4.1.5 Retrieving Disk Quota Information

Frequently, it is important to limit the amount of disk space certain users consume. The VMS Disk Quota Utility (DISKQUOTA) gives the system manager this capability. Users who have READ access to the quota file can enter the SHOW QUOTA command to determine how much disk space any user on the system has been allocated. Users who do not have READ access to the quota file can use the SHOW QUOTA command to determine their own allotments.

You can manage the checking on a per-volume basis at mount time by using the MOUNT qualifier /[NO]QUOTA. (You can also use DISKQUOTA to allow or disallow amounts of disk space.) You must have the VOLPRO user privilege, or your UIC must match the UIC written on the volume, in order to use the MOUNT command qualifier /QUOTA.

Enter the DCL command SHOW QUOTA to find out whether a quota exists for any specific user on a specific disk. The display that results from the SHOW QUOTA command gives the quotas used, authorized, and available.

A user can have a quota for any disk volume on the system. In some cases, the user is permitted a certain authorized limit plus an overdraft limit. Generally, only the authorized limit applies. However, certain system programs, such as editors, can employ the overdraft feature when the authorized limit is exceeded.

If you run out of disk space during the creation of a file, you receive a system message. If you cannot obtain sufficient space by purging or deleting unnecessary files, you may need to contact the system manager to increase your disk quota. If you attempt to write a file to a spooled printer, you must have WRITE access and have sufficient quota on the disk associated with that printer.

Examples

1
```
$ SHOW QUOTA
User [DOCUMENTATION,MALCOLM] has 2780 blocks used, 7220 available,
of 10000 authorized and permitted overdraft of 500 blocks on DISK$
```

The SHOW QUOTA command displays the amount of disk space authorized, used, and still available on the current default disk for the present user. The permitted overdraft in this example is 500 blocks.

```
2  $ SHOW QUOTA /USER=[DOCUMENTATION,JONES]/DISK=XXX1:
   %SYSTEM-F-NODISKQUOTA, no disk quota entry for this UIC
```

This SHOW QUOTA command displays the fact that the user with UIC [DOCUMENTATION,JONES] has no disk quota allocation on device XXX1.

```
3  $ SHOW QUOTA /USER=[DOCUMENTATION,ELAINE]
   User [DOCUMENTATION,ELAINE] has 27305 blocks used, 2305 OVERDRAWN,
   of 25000 authorized and permitted overdraft of 4000 blocks on DISK$
```

This SHOW QUOTA command illustrates a user with an overdrawn quota.

4.2 Using DCL to Modify File Characteristics

DCL provides commands that enable you to establish and modify the characteristics of files residing on disk and magnetic tape volumes. The following commands establish or modify file characteristics:

- SET ACL
- SET DIRECTORY (disk only)
- SET FILE (disk only)
- SET MAGTAPE (magnetic tape only)
- SET PROTECTION (disk only)
- SET VOLUME (disk only)

Each of these commands is discussed in detail in the *Guide to Using VMS Command Procedures*. For a complete list of the command qualifiers and parameters applicable to each of these DCL commands, see the *VMS DCL Dictionary*.

4.2.1 Modifying Directory Characteristics

The DCL command SET DIRECTORY modifies the characteristics of one or more directories. The following examples illustrate two uses of the SET DIRECTORY command.

Examples

```
1  $ SET DIRECTORY/VERSION_LIMIT=5/CONFIRM [SMITH.FORTRAN]
```

In this example, the SET DIRECTORY command limits the number of versions to five for files created after the command is issued. The /CONFIRM qualifier requests that you confirm whether or not the specified directory should actually be modified.

```
2  $ SET DIRECTORY/OWNER_UIC=[DOCUMENTATION,GRAY] [DAVIDSON],[USERS]
```

Here, the SET DIRECTORY command modifies both the [DAVIDSON] and [USERS] directories, changing their owner UICs. Use of the /OWNER_UIC qualifier requires SYSPRV or GRPPRV for changing the ownership at the system or group level.

4.2.2 Modifying Disk File Characteristics

Use the DCL command SET FILE to modify the characteristics of one or more files. The examples that follow illustrate three ways the SET FILE command can be used to modify file characteristics.

Examples

1
```
$ SET FILE/EXPIRATION_DATE=15-APR-1988:11:00 BATCH.COM;3
```

This SET FILE command requests that the expiration date of the file BATCH.COM;3 be set to 11:00 A.M., April 15, 1988.

2
```
$ SET FILE/BEFORE=15-APR-88/ERASE_ON_DELETE PERSONNEL*.SAL
```

This SET FILE command calls for all files that match the file specification PERSONNEL*.SAL and that are dated before April 15, 1988. Disk locations are erased for files that are deleted with commands such as DELETE or PURGE.

3
```
$ SET FILE/OWNER_UIC=[DOCUMENTATION,GRAY]/VERSION_LIMIT=100 MYFILE.DAT
```

This SET FILE command modifies the characteristics of the file MYFILE.DAT, changing the owner UIC and assigning a file version limit of 100. Note that the /OWNER_UIC qualifier requires SYSPRV or GRPPRV for changing the ownership at the system or group level.

4.2.3 Modifying Magnetic Tape Device Characteristics

Use the DCL command SET MAGTAPE to define the default characteristics associated with a specific magnetic tape device for subsequent file operations. The device must not be currently allocated to any other user.

The following examples illustrate uses of the SET MAGTAPE command in conjunction with the MOUNT command.

Examples

1
```
$ MOUNT MTB1:/FOREIGN
$ SET MAGTAPE MTB1: /DENSITY=800
```

The MOUNT command mounts a foreign tape on the device MTB1. The SET MAGTAPE command defines the density at 800 bpi for writing to the magnetic tape. (The density is reset only if the magnetic tape has never been written before.)

2
```
$ MOUNT MTA0:/FOREIGN
$ SET MAGTAPE MTA0:/SKIP=FILES:4
```

The MOUNT command mounts a foreign magnetic tape on the device MTA0; the SET MAGTAPE command directs the I/O subsystem to position the magnetic tape to skip four files.

3
```
$ MOUNT MTA1:/FOREIGN
$ SET MAGTAPE/REWIND MTA1:
```

The MOUNT command mounts a foreign tape on the device MTA1; the SET MAGTAPE rewinds the volume.

4.2.4 Modifying File Protection Characteristics

Use the SET PROTECTION command to change or reset the protection characteristics for one or more files. If you include a protection code, the file protection is changed to that specified in the code. When you omit the protection code and do not use the /PROTECTION qualifier, the file protection changes to the default file access established by the SET PROTECTION/DEFAULT command. See the SET PROTECTION/DEFAULT command for information on how to change the default file protection.

All disk and magnetic tape volumes have protection codes that restrict access to the volume. The protection codes for magnetic tape volumes are assigned with the INITIALIZE and MOUNT commands. Protection characteristics on magnetic tape volumes cannot be changed by the SET PROTECTION command.

For disk volumes, each file on the volume can have a different protection associated with it. The SET PROTECTION command and other file-manipulating commands allow you to define the protection for individual files.

If you omit both the code and the /PROTECTION file qualifier, your current default protection (established by the SET PROTECTION/DEFAULT command) is applied to the file.

The examples that follow illustrate using the SET PROTECTION command.

Examples

1
```
$ DELETE INCOME.DAT;3
%DELETE-W-FILNOTDEL, error deleting DISK1:[SMITH]INCOME.DAT;3
-RMS-E-PRV, insufficient privilege or file protection violation
$ SET PROTECTION=OWNER:D INCOME.DAT;3
$ DELETE INCOME.DAT;3
```

The file INCOME.DAT;3 has been protected against deletion. This SET PROTECTION command changes only the owner's DELETE access for the file INCOME.DAT;3. Now the file can be deleted.

2
```
$ SET PROTECTION -
_$PAYROLL.LIS/PROTECTION=(SYSTEM:R,OWNER:RWED,GROUP:RW),-
_$PAYROLL.OUT/PROTECTION=(SYSTEM:RWED,GROUP:RWED,WORLD)
```

In this example, the SET PROTECTION command changes the protection codes applied to two files. To the file PAYROLL.LIS, it gives the system READ access, the owner READ, WRITE, EXECUTE, and DELETE access, and users in the owner's group READ/WRITE access. To the file PAYROLL.OUT, it gives the system and group all types of access; the current access for owner does not change, but world is denied all types of access.

```
3   $ SET PROTECTION A.DAT, B.DAT/PROTECTION=OWNER:RWED, C.DAT
```

The SET PROTECTION command specifies that the file A.DAT should receive the default protection established for the owner's files. The existing protection for the file B.DAT is overridden, only for the OWNER category, to provide READ, WRITE, EXECUTE, and DELETE access. Note that no protection is specified for the file C.DAT at either the command or file level. Thus, like A.DAT, C.DAT receives the default protection.

Since no version numbers are specified in this example, the protection settings affect only the highest versions of the three files.

```
4   $ SET PROTECTION=OWNER:D -
_$ [MALCOLM.SUB1]SUB2.DIR/PROTECTION=GROUP:D
```

This SET PROTECTION command changes the protection for the owner and group categories of the subdirectory [MALCOLM.SUB1.SUB2] to permit deletion. However, the protection for world and system categories is not changed.

4.2.5 Modifying User Identification Code Characteristics

Although DIGITAL discourages it, you can use the DCL command SET UIC to establish a new user identification code (UIC) as your default. To use this command, however, you need a special privilege called Change Mode to Kernel (CMKRNL).

If you have the appropriate privilege, you can use the SET UIC command to gain access to a restricted file — that is, a file contained in a directory whose protection restricts access to the owner of that directory. You can also use the SET UIC command to gain access to a restricted magnetic tape volume.

The following examples illustrate how to use the SET UIC command.

Examples

```
1   $ SET UIC [370,10]
```

This command establishes your UIC as [370,10]. You can now read or modify any files whose access is restricted to this UIC.

```
2   $ SET UIC [214,4]
    $ SET DEFAULT [ANDERSON]
```

The SET UIC command sets your UIC to [214,4]; the SET DEFAULT command sets the default directory name to [ANDERSON].

```
3   $ SET UIC [ELAINE]
```

This example sets the UIC to be that of the user named ELAINE. You can optionally include the group name in the specification. For example, you could set the UIC to [ET,ELAINE]. Note that, since the user name is unique accross the system, the group name need not be included in the SET UIC command description. However, the user's group name as well as the user name will always be included in UIC displays.

4.2.6 Modifying Volume Characteristics

Use the DCL command SET VOLUME to modify the characteristics of one or more mounted Files–11 disk volumes. In order to use this command, you must have WRITE access to the index file on the volume. If you are not the owner of the volume, you must have either a system UIC or the user privilege SYSPRV. You must then specify the name of one or more mounted Files–11 volumes.

The examples that follow illustrate how the SET VOLUME command can be used.

Examples

1
```
$ SET VOLUME/DATA_CHECK=(READ,WRITE) DBC5
```

This command requests that data checks be performed following all read and write operations to DBC5.

2
```
$ SET VOLUME/LABEL=LICENSES DBC5:
```

This command encodes the label LICENSES on the volume DBC5. Note that, if characters in labels are entered in lowercase, they are changed to uppercase by the /LABEL qualifier.

4.3 Using DCL to Access Files

In addition to executing user programs that perform I/O operations on files residing on disk and magnetic tape volumes, you can use DCL commands to access disk and magnetic tape files for read and write operations.

Note that this section describes how to use DCL commands to access disk and magnetic tape files at the file level (as opposed to the record level). Although DCL does allow you to manipulate files at the record level, performance considerations usually warrant the use of a conventional programming language. DIGITAL recommends that you write programs using the VMS Record Management Services (RMS) facilities that are specifically designed to access files at the record level. You can write these programs in VAX MACRO or in any of the higher-level languages supported by the VMS operating system.

If you wish to access disk and magnetic tape files at the file level, you can take advantage of DCL commands. As mentioned above, you cannot use DCL commands to read or write files that are not in the standard formats supported by the VMS operating system. If the file formats are not standard, the volumes on which they reside must be mounted with the /FOREIGN qualifier.

The following sections provide examples of the steps you can use to access files on disk and magnetic tape volumes. In particular, these sections describe how to manipulate disk and magnetic tape files (at the file level) for READ and WRITE access.

4.3.1 Accessing Disk Files for Read and Write Operations

You can access disk files for both read and write operations. The next three sections show how to use DCL to access disk files for both types of operations.

These sections contain examples of how to do the following:

- Read files from a mounted disk volume
- Write files to a disk volume that must first be allocated, initialized, and mounted
- Write files from a default directory to a volume that must first be allocated, initialized, and mounted on a magnetic tape device

Although the examples used in the following sections show how to access disk files on RK06/RK07 disk packs, they also apply to other devices.

4.3.1.1 Reading Files from a Disk Volume

To read the contents of a disk file, use the DCL command TYPE, which displays the contents of a file on your terminal. To find the exact location of the disk file you want to read, use the DCL command DIRECTORY.

For example, if you wish to read the contents of a file named HISFILE, which is located somewhere in the directory [CHARLES] on a disk device whose logical name is DISK$DOCUMENT, look for the exact location of HISFILE by entering the following command:

```
$ DIRECTORY DISK$DOCUMENT:[CHARLES...]HISFILE.*
```

This command instructs the VMS operating system to search the entire [CHARLES] directory, including all the subdirectories, for all file types and version numbers of HISFILE. The following information will be displayed on your terminal:

```
Directory DISK$DOCUMENT:[CHARLES.MEMO]

HISFILE.UPD;1

Total of 1 file.
```

This display informs you that there is only one version of HISFILE, that its file type is UPD, and that it resides in the [CHARLES.MEMO] directory.

To read the contents of this file, enter the following command:

```
$ TYPE [CHARLES.MEMO]HISFILE.UPD
```

The contents of HISFILE will be displayed on your terminal.

4.3.1.2 Writing Files to a Disk Volume

Before you can write files to a disk volume, the volume must be properly prepared (see Chapter 3).

Because disks are random-access devices and since files must be listed in directories, you must create a directory to contain your files on the disk volume after you have initialized it, as in the following example:

```
$ CREATE/DIRECTORY DMA3:[PUBS]
$ DEFINE P DMA3:[PUBS]
$ COPY *.*  P
$ COPY [PRIMER]*.*  P
$ COPY [COMMANDS]*.*  P
```

The CREATE/DIRECTORY command creates a directory file named [PUBS] on the device DMA3, and the DEFINE command defines the logical name P as DMA3:[PUBS]. The COPY command copies the highest versions of all files in the current default directory and in the directories [PRIMER] and [COMMANDS] to the newly created directory.

4.3.1.3 Writing Files from Disk Volumes to Magnetic Tape Volumes

The next example describes how to use DCL to write files from a default directory on a disk volume to an ANSI-labeled magnetic tape volume. It includes examples that show how to allocate, initialize, and use a magnetic tape to copy a set of your disk files. The procedures are similar to those for copying files from disk volumes to disk volumes. One main difference, however, is that magnetic tapes are sequential-access devices and do not have directories. (The characteristics of magnetic tape files are described more thoroughly in the next section.)

First, allocate a drive as follows:

```
$ ALLOCATE MT:  TAPE_DEVICE
%DCL-I-ALLOC _MARS$MTA2: allocated
```

This ALLOCATE command requests the allocation of a magnetic tape drive whose name begins with MT. TAPE_DEVICE is a logical name, which in this case refers to MTA2.

The system response indicates that unit 2 on controller A was available and is now allocated to you. You can now physically load the magnetic tape on the drive. Be sure the write ring on the magnetic tape is in place; if it is not, you cannot write to the magnetic tape.

Next, initialize the magnetic tape by entering the following:

```
$ INITIALIZE  TAPE_DEVICE:  GMB001 -
_$ /PROTECTION=(GROUP:R,WORLD)
```

The INITIALIZE command specifies the logical name for the volume (TAPE_DEVICE, which in this case refers to MTA2) and the volume label for the magnetic tape volume (GMB001). The label can be no longer than six characters. The /PROTECTION qualifier defines a protection code restricting GROUP access to read and allowing no WORLD access. You can now enter the MOUNT command to mount the volume and write files to it, as in the following example:

```
$ MOUNT  TAPE_DEVICE:  GMB001
%MOUNT-I-MOUNTED, GMB001 mounted on _MTA2:
$ COPY *.*  TAPE_DEVICE:
```

The MOUNT command specifies the device name and volume label of the volume on the device. The COPY command copies the highest versions of all files in your default directory onto the magnetic tape. The file names, file types, and version numbers of the output files default to the same file names, file types, and version numbers as the input files.

If you enter the COPY command with the /LOG qualifier, the system will send a message to the current SYS$OUTPUT device after each file has been copied. You can also use the DIRECTORY command to verify that the files were successfully copied.

```
$ DIRECTORY TAPE_DEVICE:
```

This DIRECTORY command lists the file names and file types of all files on the magnetic tape.

When you have finished using the magnetic tape, dismount and deallocate it as follows:

```
$ DISMOUNT TAPE_DEVICE:
$ DEALLOCATE TAPE_DEVICE:
```

If you do not dismount and deallocate the magnetic tape, the system does so automatically when you log out.

4.3.2 Accessing Magnetic Tape Files for Read and Write Operations

When you request access to an ANSI-labeled volume or a file, the VMS operating system checks at the volume and file level to ensure that your process can access the volume or file. The level at which the system checks access depends on the operation you request and the type of access the operation requires.

When you access a volume or a file, the VMS software reads the volume- and file-header labels to determine whether access to the volume or file is restricted. Which label is read depends on the operation requested. For example, if you want to mount a volume, your process must have access to it. Thus, the operating system reads the volume labels only.

This section describes file-level access for magnetic tapes. For more detailed information on volume-level access and protection requirements for magnetic tapes, see Chapter 2.

Your access to a particular file is determined by the protection that has been set on that file. The expiration date field in the header can prevent you from overwriting or appending to a file immediately preceding the one in question. If the expiration date field has not been reached, the file has not expired. You cannot overwrite an unexpired file unless you specify the /OVERRIDE=EXPIRATION qualifier when you mount the volume.

The manner in which the file is accessed to perform an operation is called the access type. READ or WRITE access, or both, are required.

Before accessing a particular file for a read or write operation, you may want to search the magnetic tape volume for that file. The following section describes how to use the DIRECTORY command to locate a file or group of files on a magnetic tape volume.

4.3.2.1 Locating ANSI-Labeled Magnetic Tape Files for READ or WRITE Access

When you specify a VMS or ANSI file name for a file residing on magnetic tape, the magnetic tape file system compares the file name with the file header labels of each file until it finds a match in the file identifier field of the file header labels.

If you supply a version number in the file name, it is compared with the generation number and generation version-number fields in the first file header label. If you do not specify a version number, the magnetic tape file system neither defaults a version number nor checks the generation number and generation version-number fields. The magnetic tape file system selects the first file on the magnetic tape whose file name in the file identifier field matches the specified file name.

Neither the directory nor the latest version number concept is supported by the VMS operating system for magnetic tape volumes. VMS does not search for or list the latest version of a specified file. The magnetic tape file system cannot increment version numbers of files written to magnetic tape; therefore, two or more files in the same volume set can have the same file name and version number.

Because the magnetic tape file system selects the first matching file name and version number (if specified), the position of the magnetic tape within the volume set determines which file is returned on a search operation. A search operation begins at the current position, so you may want to rewind the volume set before accessing a file.

The search for a matching file and version number (if specified) continues at the beginning of the header-label set of the next file. The search ends when the magnetic tape is positioned at the file where the search began. If the requested file is not found on the current volume, the remaining volumes in the volume set are searched sequentially, according to their relative volume numbers, until either a file name match occurs or the entire volume set is searched.

If a file name match occurs, the internal file identification number is constructed from the file section number, file sequence number, and relative volume number. Although you can access a disk file by its file identification number, you cannot access a magnetic tape file this way.

The VMS operating system does support the use of wildcard characters in file specifications for magnetic tape volumes. However, there are certain restrictions on using wildcard characters with magnetic tape files. Unlike files on disk volumes, which support the asterisk (*), percent sign (%), ellipsis (. . .), and minus sign (-) characters, magnetic tape volumes support only the asterisk and percent-sign wildcard characters with VMS file names. ANSI file names support only the asterisk wildcard character.

The asterisk wildcard character matches file specifications by field or portion of a field. The percent sign wildcard character matches any character in a file specification only by character positions within a field.

With VMS file names, you can specify the asterisk and the percent sign anywhere in the file name and file type field to match file name specifications by field or character position within a field. You cannot use the percent sign in the version number field. Only the asterisk wildcard character can be used in the version number field.

With ANSI file names, a single asterisk in a field is the only wildcard character that can be used. ANSI file names do support the special set of ASCII "a" characters. Unlike VMS file names, which can consist of up to 39 characters each for the file name and file type, ANSI file names can have a maximum of 17 characters in length. Whether you choose VMS file names or ANSI file names depends on the type of applications you want to perform.

The examples that follow illustrate how to use wildcard characters in file specifications to search for files on magnetic tape volumes. These examples also show how the DIRECTORY command can be used in conjunction with magnetic tapes. Note that the DIRECTORY command does not work the same with magnetic tape files as with disk files.

Examples

1
```
$ DIRECTORY MFA1:*.*;*
```

This command instructs VMS to search a volume set. Because asterisks are used in the file specification and the asterisk is a valid wildcard character for both ANSI and VMS file names, both VMS and ANSI file names will be returned. Note that ANSI file names will be returned within quotation mark characters.

2
```
$ DIRECTORY MTA1:%*.*;*
$ DIRECTORY MTAO:*.%*;*
```

In these two commands, the search can only match with VMS file names because the percent sign is not valid for ANSI file names. In the second command, the file type field must contain at least one character. Files with no file type are not returned.

3
```
$ DIRECTORY MTAO:*.;*
```

In this example, the DIRECTORY command instructs VMS to search for files with ANSI file names, as well as VMS file names that have a null file type.

4.3.2.2 Reading Files on Magnetic Tape Volumes

When a magnetic tape file is accessed for a read operation, the magnetic tape is positioned at the beginning of the file section after the file header labels.

You can use the DCL command TYPE to read a file or group of files on the magnetic tape volume and to display the contents of the file on your terminal. For example, if you want to read the contents of a file named TESTFILE.DOC;1 (which you know from your above directory searches is a VMS file residing on the magnetic tape device MTA1), enter the following command:

```
$ TYPE MTA1:TEST*.%*;*
```

You will then receive the following display on your terminal:

```
MTA1:TESTFILE.DOC;1

This is a test file.
```

When a file residing on a magnetic tape volume is accessed only for reading the attributes in the header labels (rather than the data in the file section), the magnetic tape file system returns the RMS attributes to your process. For example, when you specify the DIRECTORY/FULL command for a volume, file, or list of files, the magnetic tape file system selects the file identifiers from

the header labels, returns the file attributes to your process, and positions the magnetic tape after the header labels of the last file accessed.

A magnetic tape file opened for read access is closed either implicitly or explicitly. The file is closed implicitly when the driver encounters a tape mark while a file is being read. The magnetic tape file system then reads the trailer labels, closes the file, and positions the magnetic tape at the next file.

The file is closed explicitly when you deaccess the file before all the data in the file is read. The magnetic tape file system then closes the file without reading the trailer labels, and the magnetic tape remains at the current position.

4.3.2.3 Writing to Files on Magnetic Tape Volumes

When you use DCL to access an existing file for a write operation, one of the following operations is actually being performed: append or update. The difference between append and update write operations can be explained in the following way:

- Append Access

  When a file is accessed for an append operation, the magnetic tape is positioned at the EOF before the tape mark that precedes the trailer labels. After the file is appended and closed, all files beyond the appended file are lost. When the positioning is complete, the processing is handled as if the file had been created as described in the section on file creation.

- Update Access

  When a file is accessed for an update operation, the magnetic tape is positioned at the BOF section after the header labels. After the file is written to and closed, all files beyond the updated file are lost. The processing is handled as if the file had been created.

Note that you can update or append magnetic tape files only when the header label contains a value of zero for the buffer offset length. For more information on how to update and append magnetic tape files, see Chapter 5.

In addition to updating and appending files on a magnetic tape volume, you can access a volume for a write operation by using the CREATE command to write a new file to the magnetic tape volume. For example, you can enter the following command string:

```
$ CREATE MTA0:MYFILE
```

You can then write the contents of the file, without leaving the DCL command level, before closing the file.

If you do not specify the /OVERRIDE=EXPIRATION qualifier, the magnetic tape file system checks the expiration date field on the file before it allows you to write to that file. When more than one file is to be overwritten, the magnetic tape file system also checks the expiration date of the file immediately following the file to be overwritten. For example, before you append to a file, the magnetic tape file system checks the expiration dates of both the file being appended and the file immediately following. If the expiration date of either file has not been reached, the magnetic tape file system does not allow you to append the file.

When files are written to a magnetic tape volume, the magnetic tape file system performs access checks, writes labels, and, if necessary, switches volumes. If the new file will overwrite an existing file, the magnetic tape file system checks the expiration date and accessibility fields of the existing file. If overwriting is allowed, the magnetic tape file system overwrites the header label set of the existing file, creates the file section, writes the trailer labels, and writes two tape marks to denote the logical end-of-volume (EOV). All files following the newly created file are lost.

To close a magnetic tape file that was opened for WRITE access, the magnetic tape file system issues commands to the driver to write the labels, which are followed by a double tape mark that indicates the logical EOV.

4.4 Using Command Procedures to Access Foreign Volumes

The command procedures in this section allow you to create, read, or write magnetic tape data in a simple, user-defined, foreign format.

```
$!
$!              FOREIGN.COM
$!
$! This is the master command procedure. It sets up the user account and
$! mounts the volume with the /FOREIGN qualifier. If the user wants to
$! read a foreign volume, the FORREAD.COM command procedure is called.
$! If the user wants to write a foreign volume the FORWRITE.COM
$! command procedure is called.
$!
$       verify_off_on = F$VERIFY ( 0 )
$       ON CONTROL_Y THEN GOTO clean_up
$       ON WARNING THEN GOTO clean_up
$
$!
$! If VOLPRO privilege is not set but the user account has SETPRV, VOLPRO is
$! set to allow the user to mount a new unformatted volume.  If VOLPRO
$! privilege cannot be set, the user process is notified that the process
$! has insufficient privilege to write an unformatted volume. The user is also
$! asked whether to continue or exit the procedure.
$!
$       volpro_off_on = F$SETPRV ( "VOLPRO" )
$       IF F$PRIVILEGE ( "VOLPRO" ) THEN GOTO cont
$       WRITE SYS$OUTPUT "Insufficient Privilege to write an unformatted volume!"
$       INQUIRE/NOPUNC continue "Do you wish to continue (Y/N) ? "
$       IF .NOT. continue THEN GOTO clean_up
$ cont:
$
$!
$! Find out where the volume is mounted
$!
$ get_drive:
$       INQUIRE tape_drive "Tape drive"
$       IF tape_drive .EQS. "" THEN GOTO get_drive
$       tape_drive = tape_drive-":"+":"
$       IF .NOT. F$GETDVI (TAPE_DRIVE,"EXISTS")
$               THEN GOTO NOSUCHDEV
$       ASSIGN 'tape_drive' tape
$
```

```
$!
$! Try allocating and mounting the volume
$!
$       ALLOCATE tape:
$       MOUNT/NOASSIST/FOREIGN/OVERRIDE=(ACCESSIBILITY,EXPIRATION) tape:
$
$!
$! The user is asked whether a file will be read or written
$!
$
$ read_write:
$       INQUIRE/NOPUNC operation "Read or Write a file ? "
$       IF F$LOCATE ( operation, "READ" )  .EQ. 0 THEN GOTO read_op
$       IF F$LOCATE ( operation, "WRITE" ) .EQ. 0 THEN GOTO write_op
$       GOTO read_write
$
$ read_op:
$       @FORREAD
$       GOTO clean_up
$
$ write_op:
$       @FORWRITE
$
$ clean_up:
$
$!
$!  Reset the user account the way it was before the command procedure began
$!
$       SET NOON
$       DISMOUNT/NOUNLOAD tape:
$       DEALLOCATE tape:
$       DEASSIGN tape
$       volpro_off_on = F$SETPRV ( volpro_off_on )
$       verify_off_on = F$VERIFY ( verify_off_on )
$
$ EXIT
$
$ NOSUCHDEV:
$        WRITE SYS$OUTPUT "No Such Device"
$        GOTO get_drive

$!  FORWRITE.COM    Writes Data to a Foreign Volume
$!
$!  This command procedure writes data to a foreign volume.
$!  The data format is as follows:
$!       Each record is a block.
$!       Records are variable length.
$!
$!      The first four characters of a record are digits that are
$!      padded on the left with spaces. This sequence field starts at
$!      1 and increases by 1 with each record.
$!
$!      The fifth character of a record is a comma.
$!
$!      The sixth through ninth characters are digits that are padded on the
$!      left with spaces. This is a size field, which is the size of the data
$!      field.
$!
$!       The tenth character of a record is a vertical bar.
$!
$!       The rest of the record is data.
$!
$!       Records are padded to be at least 20 characters long.
```

```
$!
$
$       seq_num = 0    ! initialize the sequence number
$       spaces  = "           "   ! used to pad records less then 20 chars
$
$       OPEN/WRITE tape_file tape: ! open the output file
$ next_line:
$
$       seq_num = seq_num + 1  ! increment the sequence number
$
$!  Prompt the user for the data
$!       close the file and exit if ^Z is entered
$!
$       READ/PROMPT="Record # 'seq_num' : "/END_OF_FILE=end_of_input -
            SYS$COMMAND user_data
$
$!  Find the size of the record and the number of pad characters needed
$!  Note That a negative number of pad characters does not return any characters
$!
$       data_size = F$LENGTH ( user_data )
$       pad_chars = 10 - data_size
$       IF pad_chars .LT. 0 THEN pad_chars = 0
$
$!  Construct the output record using FAO
$!
$       out_rec = F$FAO ( "!4UL,!4UL,!AS", seq_num, data_size, user_data ) + --
           F$EXTRACT ( 1, pad_chars, spaces )
$
$       WRITE tape_file out_rec ! write the formatted output record
$       GOTO next_line
$
$ end_of_input:
$       CLOSE tape_file


$!             FORREAD.COM    Reads Data from a Foreign Volume
$!
$!  This command procedure is called when data will be read.  The procedure
$!  reads data on a foreign volume.  The data format is defined in
$!  FORWRITE.COM command procedure.
$!
$
$       seq_num = 0    ! initialize the sequence number
$
$       OPEN/READ tape_file tape: ! open the output file
$ next_line:
$
$        seq_num = seq_num + 1  ! increment the sequence number
$
$        READ/END_OF_FILE=end_of_input tape_file in_rec
$
$        record_num = F$EXTRACT ( 0, 4, in_rec ) -- " " -- " " -- " "
$        record_num = F$INTEGER ( record_num )
$        IF seq_num .NE. record_num THEN -
           WRITE SYS$OUTPUT "Error possible data lost, record sequence broken!"
$
```

```
$!
$!  Find the size of the data and extract the data
$!
$       data_size = F$EXTRACT ( 5, 4, in_rec ) -- " " -- " " -- " "
$       data_size = F$INTEGER ( data_size )
$       data_rec  = F$EXTRACT ( 10, data_size, in_rec )
$
$       WRITE SYS$OUTPUT data_rec ! write the data output record
$       GOTO next_line
$
$ end_of_input:
$       CLOSE tape_file
```

5 Transferring Information

The VMS operating system provides various facilities for transferring information contained on disk and magnetic tape media. In particular, the DCL command COPY and the VMS Convert and Exchange Utilities can be used to transfer disk and magnetic tape information.

5.1 Transferring Information Within and Across Operating Systems

In many cases, you will be able to transfer information without physically transporting media. You may, however, find it necessary to transfer files between systems that are not connected by a communications link. Under these circumstances, you must be able to physically move your files from one location to another. A convenient way to do this is to copy your files to a portable volume, such as a magnetic tape reel or disk pack, and then carry that volume to the location of the other system.

The VMS operating system supports the transfer of information contained on disks and magnetic tapes both within the VMS system and across other operating systems. It provides a number of facilities to assist you in both types of information transfer. The two most frequently used facilities for transferring information are the DCL command COPY and the VMS Exchange Utility (EXCHANGE).

In many cases, you will find the COPY command and the Exchange Utility sufficient for accomplishing information transfers. You may also find the VMS Convert and Analyze/RMS_File Utilities useful for transferring information, especially in foreign-volume or non-file-structured environments.

Under some circumstances, you will need to use the Backup Utility (BACKUP) to transfer files. When using magnetic tape, for example, BACKUP is the only means of transferring entire directory trees or files that are not sequentially structured.

Each of the information transfer facilities provided by the VMS system, except BACKUP, is described in the sections that follow. (See the *VMS Backup Utility Manual* for information on the use of BACKUP.)

5.2 Using the COPY Command to Transfer Information

One way of transferring information on disk or magnetic tape media is to use the DCL command COPY. The COPY command copies files from

- Disk to disk
- Disk to magnetic tape
- Magnetic tape to disk
- Magnetic tape to magnetic tape

The sections that follow describe how the COPY command is used with both disk and magnetic tape files.

5.2.1 Copying Files from Disk Volumes

This section describes how to use the COPY command to copy files from disk volumes to other disk volumes and from disk volumes to magnetic tape volumes.

The default format for files on disk volumes is called Files–11 Structure Level 2. You can also initialize disks in the Files–11 Structure Level 1 format, which is the format used by other DIGITAL operating systems including RSX–11M, RSX–11M–PLUS, RSX–11D, and IAS.

In the following example, the COPY commands copy the highest versions of all files in the current default directory and in the directories [PRIMER] and [COMMANDS] to the directory [PUBS]:

```
$ DEFINE P  DMA3:[PUBS]
$ COPY *.*  P:
$ COPY [PRIMER]*.*  P:
$ COPY [COMMANDS]*.*  P:
```

If you want to copy files from a disk directory — such as your default directory — on a public disk volume to a private Files–11 disk volume, you can also use the COPY command. Before copying these files, however, you must set up (allocate, initialize, and mount) a disk device.

In the following example, assume that the disk device DMA5 has been allocated to your process and that a disk volume has been initialized and mounted on that device. Also assume that you have a directory called PRIVATE already created on that volume. Copy the highest version of all the files in your default directory to the directory on that volume by entering the following command:

```
$ COPY *.*  DMA5:[PRIVATE]
```

You can also use the COPY command to copy files from a disk volume to a magnetic tape volume. The procedure for copying files from disk volumes to magnetic tape volumes is similar to those previously outlined for copying files across disk volumes. Note that magnetic tapes are sequential-access devices and do not have directories.

Again, you must set up (allocate, initialize, and mount) a magnetic tape device before copying disk files to the magnetic tape volume. In the following example, assume that MTA2 has been allocated to your process and that a magnetic tape volume has been initialized and mounted on that device. You can now use the COPY command to write files to the magnetic tape volume, as in the following example:

```
$ COPY *.*  MTA2:
```

In this case, the highest versions of all files in your default disk directory are copied to the magnetic tape volume on MTA2. The file names, file types, and version numbers of the output files default to the same file names, file types, and version numbers as the input files.

If you enter the COPY command with the /LOG qualifier, the system sends a message to the current SYS$OUTPUT device after each file has been copied. To verify that the files were successfully copied, use the DIRECTORY command. For example, the following command lists the file names and file types of all files on the magnetic tape volume:

```
$ DIRECTORY MTA2:
```

When you copy files from disk to ANSI-labeled volumes, the following items are not preserved:

- Directory specifications
- Individual file protection
- User identification code (UIC)
- Creation time (but the date is preserved)
- Revision and backup dates and times

5.2.2 Copying Files from Magnetic Tape Volumes

This section describes how to use the COPY command to copy files from magnetic tape.

The default format for files on magnetic tapes is the ANSI-labeled volume. The VMS system supports sequential, relative, and indexed files on disks, but only sequential files can be copied to ANSI-labeled volumes. The only valid record formats are variable-length (ANSI D) and fixed-length (ANSI F). In the example that follows, the ANSI file NEWINFO is copied from magnetic tape volume to the file TESTING.DAT:

```
$ COPY MTA1:NEWINFO    TESTING.DAT
```

Although the VMS system supports stream and variable with fixed-length control (VFC) records, it encodes these records in a variable-length format on ANSI-labeled volumes. Non-VMS systems do not distinguish stream records from VFC records; instead, they interpret both as variable-length records. Therefore, neither stream nor VFC records should be created on volumes that will be used for information interchange to a non-VMS system.

The full set of ASCII "a" characters are supported only for ANSI-labeled volumes, not for disk volumes. Therefore, when you copy files with ANSI file names from magnetic tape to disk, specify a standard VMS file name for the output file name specification.

If you do not specify a VMS file name on output, your process receives the following error message:

```
RMS-F-FNM, error in file name
```

This message indicates that the ANSI file name is not a valid file name specification.

The entire set of Files–11 file names is supported for magnetic tapes. You can copy a disk file with the following file name to a magnetic tape volume, without having to modify the file name:

THIS_IS$A_VAXVMSLONG_FILE.LONG_EXT

You can also use the Exchange Utility to perform file transfers and format conversions for DOS–11 magnetic tape volumes, Files–11 volumes, and RT–11 block-addressable volumes. Refer to Section 5.4 of this manual or to the *VMS Exchange Utility Manual* for more information.

5.2.2.1 Continuing the Copy Command at End-of-Tape

When you are copying to or from a magnetic tape and that tape reaches the end, the system suspends processing and sends a request to you to mount the next magnetic tape in the volume set. An Operator Communication Facility (OPCOM) message similar to the following may be displayed at your terminal:

```
%%%%%%%%%%  OPCOM, 14-JUN-1985 15:23:31.78  %%%%%%%%%%
request 3, from user PLAW
MOUNT new relative volume 2 (DWOQT2) on MTA1:
```

Note: Normally you do not see this message (messages may be sent only to the operator's terminal that has been enabled for tape messages), and you may not realize that another tape is needed to complete the read or write operation.

See the *Guide to Maintaining a VMS System* for more information on OPCOM messages.

If automatic volume switching is disabled or if the magnetic tape file system cannot mount a given volume, you may need to mount a continuation volume in a volume set. See Section 3.4.4.2 for information on mounting a continuation volume. After loading the continuation volume on the drive specified in the mount request, mount the volume by entering the REPLY command with one of the following three qualifiers:

- /TO=request_id [volume identifier] — Used for both read and write copy operations. During a write operation, use the /TO qualifier if you want the volume identifier specified in the mount request to be written on the continuation volume.

  For example, to respond to the mount request 3, mount volume DW0QT2 on drive MTA1 and enter one of the following commands:

  ```
  $ REPLY/TO=3
  $ REPLY/TO=3 "DWOQT2"
  ```

  The first REPLY command does not specify a volume identifier; the second does.

- /INITIALIZE_TAPE — Used for write operations if the volume identifier on the continuation volume does not match the one specified in the mount request. The file system reinitializes the tape and mounts the volume with the new volume identifier. The magnetic tape file system then performs access checks and initializes the volume as if the INITIALIZE command had been specified. Any data on the tape prior to specifying the /INITIALIZE_TAPE command is lost. Either of the following REPLY commands is valid:

  ```
  $ REPLY/INITIALIZE_TAPE=3
  $ REPLY/INITIALIZE_TAPE=3 "DWOQT2"
  ```

  The first command does not specify a volume identifier; the second does.

- /BLANK_TAPE — Used to write to an unformatted volume. This qualifier initializes the volume and requires the VOLPRO and OPER privileges to avoid a runaway tape or timeout condition (see Chapter 3). Either of the following REPLY commands is valid:

  ```
  $ REPLY/BLANK_TAPE=3
  $ REPLY/BLANK_TAPE=3 "DWOQT2"
  ```

The first command does not specify a volume identifier; the second does. Specifying the volume identifier in either the MOUNT command or the REPLY/TO command is essential during write operations because it ensures that the correct volume is mounted on the drive and links the continuation volume to the volume set.

You can omit the volume identifier with the REPLY/TO command under two circumstances: (1) when reading from tape, the volume identifier is optional; (2) during a write operation, you must omit the volume identifier to preserve the accessibility code on a volume. If you initialize and mount a volume set in which each volume has a unique accessibility character that you want to maintain, avoid using the volume identifier since it causes the accessibility character of the first volume in the set to overwrite the accessibility code on the continuation volume. For example, to preserve the accessibility character, enter the following command, where 3 is the request identification number:

```
$ REPLY/TO=3
```

Once it receives the REPLY command, the magnetic tape file system performs checks on the continuation volume to ensure that it is the correct volume. As long as it is the correct volume with proper access codes, the system mounts the volume and reissues pending read or write requests to the continuation volume. If the volume fails any of these access checks, the volume is not mounted (or initialized and mounted, in the case of a blank tape).

The following examples illustrate ways of copying files to and from magnetic tape volumes.

Examples

1

```
$ COPY/LOG MTA1:"%&*?!SKI! """ SEASON.DAT
%COPY-S-COPIED, MTA1:[]"%&*?!SKI! """.;1
copied to WRKD:[MANUAL]SEASON.DAT;1 (120 records)
```

Since the /LOG qualifier was specified, the system returns a message that confirms the file was copied as specified and informs you how many records were copied. The ANSI file %&*?!SKI!#" (# means space) is copied to the file SEASON.DAT on the default disk and directory WRKD:[MANUAL]. The file could not have been copied to disk unless the new file name was specified. The VMS software provided defaults for segments of the file specification that were not specified.

2

```
$ COPY/LOG FORTAP.DAT MTA1:"%&*?!SKI! "" "
%COPY-S-COPIED, WRKD:[MANUAL]FORTAP.DAT;1
copied to MTA1:[]"%&*?!SKI! """.;0 (120 records)
```

In this command an ANSI file name was specified as the output file specification. Note that the trailing space in the file name %&*?!SKI!##"# (where # means space) is truncated.

3

```
$ COPY/LOG VAXVMS_LONG$FILE_NAME.LONG_EXT MTA1:
%COPY-S-COPIED, WRKD:[MANUAL]VAXVMS_LONG$FILE_NAME_EXT;1
copied to MTA1:VAXVMS_LONG$FILE_NAME.LONG_EXT;1 (80 records)
```

In this example, a VMS long file name with a long extension is copied to the volume MTA1 with the same file name and type that had been on the disk volume.

```
4  $ COPY %%.JOU;* MTA1:*.*
   %COPY-S-COPIED, WRKD:[MANUAL]C6.JOU;1 copied to MTA1:[]C6.JOU;1 (4 records)
```

In this example, all files with a two-character file name and a file type of JOU are copied to the volume MTA1 with the same file name and type as they had on the disk volume. Version numbers are preserved.

```
5  $ COPY MTAO:*.* *
   %COPY-S-COPIED, MTAO:[]TASTETEST.DAT;1
   copied to WRKD:[FOOD]TASTETEST.DAT;1 (249 records)
   %COPY-S-COPIED, MTAO:[]ALLAT;1 copied to WRKD:[FOOD]ALALL;1 (48 rcords)
   %COPY-S-NEWFILES, 2 files created
```

In this example, neither file on the tape volume had an ANSI file name. Therefore, both files were copied to the disk volume.

```
6  $ COPY MTA1:*.* [EX]
   %COPY-S-COPIED, MTA1:[].DAT;1 copied to WRKD:[EX]TEST.DAT21 records)
   %COPY-E-OPENOUT, error opening WRKD:[EX]"%&*()!SKI! """.;1 as output
   -RMS-F-FNM, error in file name
   %COPY-W-NOTCOPIED, MTA1:[]"%&*()!SKI! """.;1 not copied
   %COPY-E-OPENOUT, error opening WRKD:[EX]"SANFRAN%%%""".;1 as output
   -RMS-F-FNM, error in file name
   %COPY-W-NOTCOPIED, MTA1:[]"SANFRAN%%%""".;1 not copied
   %COPY-S-COPIED, MTA1:[]VAXVMS_LONG$FILE_NAME.LONG_EXT;1
   copied to WRKD$:[EX]VAXVMS_LONG$FILE_NAME.LONG_EXT;1 (80 records)
   %COPY-S-COPIED, MTA1:[]C6.JOU;1 copied to WRKD:[EX]C6.JOU;1 (4 records)
   %COPY-S-NEWFILES, 2 files created
```

The COPY command string specifies that all files on the volume mounted on drive MTA1 should be copied to the current default disk and directory WRKD:[EX]. However, files with ANSI file names are not copied; VMS returns an error message to the process.

The following sections provide guidelines for interchanging volumes.

5.2.3 Copying Files to and from Non-File-Structured Volumes

The VMS operating system supports the transfer of files between file-structured and non-file-structured volumes. You can use the COPY command to transfer files in the following ways:

- From file-structured volumes to non-file-structured volumes
- From non-file-structured volumes to file-structured volumes

The next two sections contain examples of each type of transfer.

5.2.3.1 Copying Files to a Non-File-Structured Volume

This section contains a procedure for copying files from a file-structured volume to a non-file-structured, or foreign volume.

The following procedure copies three files from an ANSI-labeled volume to a foreign volume.

1 Mount the volumes involved in the copy operation as follows:

```
$ MOUNT MTA1: FRESKI
%MOUNT-I-MOUNTED, FRESKI mounted on _MTA1:
$ MOUNT/FOREIGN MTAO:
%MOUNT-I-MOUNTED, mounted on _MTAO:
```

2 Enter the COPY command string, including the full device and file name specifications in the input specification, but specify the device name only in the output specification. File names are specified on input because the volume from which files will be copied is ANSI-labeled and file-structured. However, only the device name is specified on output because the volume to which the files will be copied is foreign and is not file-structured.

```
$ COPY/LOG MTA1:PROG1.EXE, PROG2.EXE, PROG3.EXE MTA0:
%COPY-S-COPIED, MTA1:[]PROG1.EXE;1 copied to MTA0: (92 records)
%COPY-S-COPIED, MTA1:[]PROG2.EXE;6 copied to MTA0: (70 records)
%COPY-S-COPIED, MTA1:[]PROG3.EXE;2 copied to MTA0: (77 records)
%COPY-S-NEWFILES, 3 files created
```

Because no version numbers were specified for the files copied from the ANSI-labeled magnetic tape volume, the first version of each specified file the MTAACP finds on the ANSI-labeled volume will be the version of the file that is copied to the foreign volume.

When you use the /LOG qualifier with the COPY command, the system informs you which files are copied and tallies the number of records copied and the number of files created.

When you copy files as shown in this example, the structure of the resulting magnetic tape is referred to as unblocked, non-file-structured. Each file record becomes a physical record on the magnetic tape. Each file boundary is indicated on the magnetic tape by a single tape mark. Two consecutive tape marks indicate end of volume. No label, file name, or attribute information is present.

5.2.3.2 Copying Files from a Non-File-Structured Volume

This section contains a procedure for copying data segments from a non-file-structured, or foreign, volume to a file-structured ANSI-labeled volume. Note that you must specify a COPY command for each segment of data that you copy from a foreign, or non-file-structured, volume to a file-structured volume.

1 Mount the volumes as follows:

```
$ MOUNT/NOLABEL MTA1: " " COLD:
%MOUNT-I-MOUNTED, mounted on _MTA1:

$ MOUNT MTA0: FEVER SEASON:
%MOUNT-I-MOUNTED, FEVER mounted on _MTA0:
```

2 Enter a COPY command for each segment on the foreign volume to be copied to files on the file-structured volume. Because the volume to which data segments are copied is file-structured, you must specify a file name in the output file specification.

```
$ COPY/LOG COLD: SEASON:FILE1.DAT
%COPY-S-COPIED, MTA1: copied to SEASON:FILE1.DAT;1 (92 records)
$ COPY/LOG COLD: SEASON:FILE2.DAT
%COPY-S-COPIED, MTA1: copied to SEASON:FILE2.DAT;1 (62 records)
$ COPY/LOG COLD: SEASON:FILE3.DAT
%COPY-S-COPIED, MTA1: copied to SEASON:FILE3.DAT;1 (23 records)
$ COPY/LOG COLD: SEASON:FILE4.DAT
%COPY-S-COPIED, MTA1: copied to SEASON:FILE4.DAT;1 (48 records)
$ COPY/LOG COLD: SEASON:FILE5.DAT
%COPY-S-COPIED, MTA1: copied to SEASON:FILE5.DAT;1 (37 records)
$ COPY/LOG COLD: SEASON:FILE6.DAT
%COPY-S-COPIED, MTA1: copied to SEASON:FILE6.DAT;1 (10 records)
```

Since the /LOG qualifier was used with the COPY command, VMS informs you that the files you specified were copied to the appropriate device and that tallies the number of records transferred.

5.3 Using the Convert Utility to Transfer Information

This section describes how to use the VMS Convert Utility (CONVERT) to transfer information to non-file-structured, or foreign, volume. It also describes how the VMS Analyze/RMS_File Utility (ANALYZE/RMS_FILE), which is useful in determining record size, is used in conjunction with CONVERT.

There may be cases where you will be unable to transfer information to a non-file-structured, or foreign, volume by using the COPY command. For example, files containing records smaller than 14 bytes cannot be copied successfully to a foreign volume with the COPY command. However, they can be transferred by using CONVERT.

CONVERT allows you to pad and extend the record size to make it suitable for a foreign volume. The following procedure illustrates how to use the Convert Utility:

1 Mount both volumes. Specify the /RECORDSIZE and /BLOCKSIZE qualifiers in addition to either the /NOLABEL or /FOREIGN qualifier when you mount the foreign volume.

```
$ MOUNT/FOREIGN/RECORDSIZE=80/BLOCKSIZE=2400 MTA1:
%MOUNT-I-MOUNTED, mounted on _MTA1:

$ MOUNT MTA0: LATER TODAY
%MOUNT-I-MOUNTED, LATER mounted on _MTA0:
```

The VMS operating system informs you that the foreign volume is mounted on drive MTA1 and that the target volume is mounted on MTA0. The /RECORDSIZE and /BLOCKSIZE qualifiers specify the tape is to be written as 80-byte records blocked in 2400-byte blocks.

2 Check the size of the largest record within a file. To do so, use the Analyze/RMS_File Utility as shown in the following example:

```
$ ANALYZE/RMS_FILE/STATISTICS TODAY:TICKET.DAT

RMS File Statistics                    14-JUN-1988 21:33:20.07
TODAY:[]TICKET.DAT;3
```

```
FILE HEADER
   File Spec: TODAY:[]TICKET.DAT;3
   File ID: (1,1,1)
   Owner UIC: [012,141]
   Protection: System:RWED, Owner:RWED, Group:WE, World:WE
   Creation Date:    5-APR-1988 14:20:04.06
   Revision Date:   17-NOV-1988 09:10:05.19, Number: 3
   Expiration Date: none specified
   Backup Date:     none posted
   Contiguity Options:  none
   Performance Options: none
   Reliability Options: none
   Journaling Enabled:  none

RMS FILE ATTRIBUTES

   File Organization: sequential
   Record Format: variable
   Record Attributes:   carriage-return
   Maximum Record Size: 255
   Longest Record: 77
   Blocks Allocated: 0, Default Extend Size: 0
   End-of-File VBN: 1, Offset: %X'0000'
   File Monitoring: disabled
   Global Buffer Count: 0

The analysis uncovered NO errors.

ANALYZE/RMS_FILE/STATISTICS TODAY:TICKET.DAT
```

3 Extend the size of records smaller than 14 bytes to that specified in the MOUNT command and ensure that the record size you specify is sufficient. To do so, enter the following commands:

```
$ CONVERT/PAD/FDL=SYS$INPUT
$_Input: TODAY:TICKET.DAT
$_Output: MTA1:
RECORD
SIZE 80
FORMAT FIX 15   EXIT
$
```

Entering the CONVERT command string, along with the input and output parameters, pads and extends the size of records when copying them from the ANSI-labeled volume mounted on MTA0 to the foreign volume mounted on MTA1. The CONVERT command and qualifiers can pad short records to correspond to the record size specified by the MOUNT command in the first step. After you enter the command line, CONVERT prompts you for input and output parameters. At the *$_Input:* prompt, supply the specification of the file containing the records that were too short. At the *$_Output:* prompt, specify only the physical or logical device name of the foreign volume. Pressing CTRL/Z terminates the file description and causes CONVERT to run. To verify that the file was copied to the foreign volume, you can rewind the foreign volume and type the volume as shown in the steps below.

4 To rewind a foreign volume, enter the following SET MAGTAPE /REWIND command string:

```
$ SET MAGTAPE/REWIND MTA1:
$
```

5 To verify the contents of a file, enter the TYPE command as follows:

```
$ TYPE MTA1:
This is a test file.
Some records contain less than 14 bytes.
This file was copied to a foreign volume with the
RMS Convert Utility.
Any record less than 80 bytes was extended and padded to
80 bytes.
```

The TYPE command types the first data segment on the volume. If you copy multiple files to a foreign volume with the RMS Convert Utility, each file is copied to the volume in a separate data segment. Therefore, you must enter a TYPE command for each data segment copied to the volume until you reach the segment that you want to verify.

This example produces what is referred to as a blocked non-file-structured volume. Each record of the input file is placed into an 80-byte area of a 2400-byte tape block. Tape marks are used to delimit files, as in the previous example.

5.4 Using the Exchange Utility to Transfer Information

The VMS system provides a utility called Exchange, which enables you to read to or write from disk and magnetic tape media being exchanged with DIGITAL-supported, non-VMS operating systems. With the VMS Exchange Utility (EXCHANGE), you can manipulate mass-storage volumes written in formats other than those normally recognized by VMS. You can also use EXCHANGE to manipulate Files–11 files that are images of foreign volumes.

You can use EXCHANGE to transfer files between foreign volumes and VMS file-structured volumes. You can also use EXCHANGE to perform volume-specific initialization and manipulation functions on the foreign volumes. The Exchange Utility enables you to convert the format of the files, as appropriate, when transferring files between volumes with different structures.

The Exchange Utility can perform file transfers and format conversions for the following:

- DOS–11 magnetic tape volumes
- Files–11 volumes
- RT–11 block-addressable volumes

In addition to transferring files, the Exchange Utility allows you to mount and dismount foreign volumes, initialize foreign volumes, list directories of volumes, delete files from block-addressable volumes, and locate bad blocks on the volume. For further information on how to use EXCHANGE to accomplish these tasks, see the *VMS Exchange Utility Manual*.

5.4.1 Invoking and Terminating the Exchange Utility

To invoke EXCHANGE, enter the following command in response to the DCL prompt:

```
$ EXCHANGE
EXCHANGE>
```

When you receive the EXCHANGE prompt (EXCHANGE>), you are in the Exchange Utility. Once you are in this utility, you can enter any of the command strings supported for EXCHANGE. For example:

```
EXCHANGE> DIRECTORY MFA0:/VOLUME=DOS11/FULL
```

This command lists all the files on the DOS–11 magnetic tape mounted on MFA0. In this case, the magnetic tape is rewound before the files are listed.

The following example illustrates the use of the MOUNT command within the Exchange Utility:

```
EXCHANGE> MOUNT DMA1:
%EXCHANGE-I-WRITELOCK, volume is write-locked
%EXCHANGE-I-MOUNTED, volume DMA1: mounted
```

This command mounts the foreign volume that is loaded in the RK07 device DMA1, making the volume available for subsequent commands. In this case, EXCHANGE recognizes that the volume itself is write-locked, and displays the message.

See the *VMS Exchange Utility Manual* for a complete list of all the commands, qualifiers, and parameters that are supported for EXCHANGE.

To exit from EXCHANGE and return to DCL level, use the EXCHANGE command EXIT or CTRL/Z.

5.4.2 Using EXCHANGE at DCL Command Level

In addition to using EXCHANGE interactively as a utility, you can use EXCHANGE as a DCL command. To do this, append a command string to the DCL command EXCHANGE as follows:

```
$ EXCHANGE DIRECTORY DMA1:/VOLUME_FORMAT=RT11
```

This DCL command lists the directory of the RT–11 volume mounted (/FOREIGN) on DMA1. At DCL level, EXCHANGE executes the single command and returns to the DCL prompt.

At DCL level, you can process only one command at a time. Some tasks for which EXCHANGE was designed, however, require more than one EXCHANGE command. For example, the use of virtual devices requires multiple EXCHANGE commands.

Tasks requiring more than one EXCHANGE command cannot be performed at DCL level. You can, however, design a command procedure to execute more than one EXCHANGE command for a particular task. In that case, you could execute the command procedure without leaving the DCL command level. For further details on how to construct command procedures using EXCHANGE, see Section 5.5.

5.5 Using Command Procedures to Transfer Information

You can design command procedures to facilitate routine transferring of information on disk and magnetic tape media.

Two examples of command procedures are included in this section. The first example is designed to transfer information by way of the COPY facility. The second example employs the Exchange Utility to transfer the information.

5.5.1 Using a Command Procedure to Copy Files

The sample command procedure in this section can be used for a variety of purposes. This procedure can copy from disk to similar disk, from disk to dissimilar disk, and from magnetic tape to magnetic tape. Note that the command procedure mounts the volume as non-file-structured.

If you are using magnetic tape media, this command procedure will not work for multi-reel magnetic tape volume sets. Magnetic tape to magnetic tape transfer is supported for single volumes only. The magnetic tape to magnetic tape copy is accomplished using the asterisk (*) wildcard character.

The sample command procedure is as follows:

```
$ vol_priv = f$setprv("volpro")
$ Type sys$input
$ on control_y then goto cleanup
$ on warning then goto cleanup
$ !
$ get_source:
$    write sys$output "Enter drive holding the write-locked master."
$    inquire source "[CS1$DRB0:, DBxx:, DDxx:, DJxx:, DLxx:, DMxx:, -
$    DQxx:, DRxx:, DUxx:, DYxx:, MFxx:, MTxx:, MUxx:,]"
$    if source .eqs. "" then goto get_source
$    source = source -":"+":"
$ !
$ get_target:
$    inquire target "Enter drive holding the write-enabled target."
$    if target .eqs. "" then goto get_target
$    target = target -":"+":"
$    kit_type := 'f$extract(0,1,source)
$    if f$locate("CS",source) .eq. f$length(source) .and. -
$    f$locate("CS",target) .eq. f$length(target) then goto prepare
$
$ ! *Note* -- This section checks to see if the console is configured
$ !          into the system.
$ !          If it does not exist, SYSGEN will be run to configure it in
$ !
$ connect_console:
$    ! (NOTE* For 750,730 media copying CONSOLE must be configured)
$    if f$getdvi("CS:1","exists") then goto prepare
$    set noon
$    proc_priv = f$setprv("cmkrnl,cmexec")
$    run sys$system:sysgen
     connect console
$    proc_priv = f$setprv(proc_priv)
$    set on
```

```
$ !
$ prepare:
$    allocate 'source'
$    inquire start "Ready source volume ''source' for mounting and
$    press return"
$ !
$ begin:
$    allocate 'target'
$    mount/foreign 'source'
$    inquire start "Ready target volume ''target' for initialization and
$    press return"
$    volume_lbl = f$getdvi(source,"VOLNAM")
$    if (volume_lbl .eqs. "") then volume_lbl := "console"
$    init 'target' 'volume_lbl'
$    If (kit_type .eqs. "M") then goto tape_copy
$    mount/foreign 'target'
$    backup/physical/verify 'source' 'target'
$    goto copy_done
$ !
$ tape_copy:
$    mount/over:id 'target'
$    copy 'source'*.* 'target'*/log
$ !
$ copy_done:
$    dismount 'target'
$    deallocate 'target'
$    write sys$output "Copy is complete"
$    dismount/nounload 'source'
$    inquire answer "Do you want to make another copy of the Source volume?"
$    if .not. answer then goto new_source
$    write sys$output "Please place volume in the target device ''target'." "
$    goto begin
$ !
$ new_source:
$    deallocate 'source'
$    inquire answer "Do you want to make a copy using a new Source device?
$    if answer then goto get_source
$    goto finish
$ !
$ cleanup:
$    if  f$getdvi(target,"mnt") then  dismount 'target'
$    if  f$getdvi(source,"mnt") then  dismount 'source'
$ !
$ finish:
$    vol_priv = f$setprv(vol_priv)
$    if f$getdvi(target,"all") then deallocate 'target'
$    if f$getdvi(source,"all") then deallocate 'source'
$    exit
```

5.5.2 Using a Command Procedure to Exchange Information

The following command procedure is designed to exchange files between the console device and the current directory on disk. The files to be copied are assumed to be in standard format as determined by file type.

```
$ WRITE SYS$OUTPUT " Command file to copy files to/from the system"
$ WRITE SYS$OUTPUT " console storage medium and the current directory."
$ WRITE SYS$OUTPUT " "
$ INQUIRE MOUNT "Is system console storage medium mounted (Y/N)?"
$ IF MOUNT THEN GOTO MOUNTED
$ WRITE SYS$OUTPUT "Please place the system console medium in the console drive"
$ INQUIRE MOUNT "and type [RET] when ready"
$ RUN SYS$SYSTEM:SYSGEN
CONNECT CONSOLE
$ MOUNT/SYSTEM/FOREIGN CSA1: "VAX console"
$ !
$ MOUNTED:
$ INQUIRE DIR "Copy from console medium (Y/N)?"
$ IF DIR THEN GOTO FROMCON
$ INQUIRE SOURCE "Enter file name(s)"
$ IF SOURCE .EQS. "" THEN GOTO EXIT
$ EXCHANGE COPY /LOG 'SOURCE' CSA1:
$ GOTO EXIT
$ !
$ FROMCON:
$ INQUIRE SOURCE "Enter console file name"
$ IF SOURCE .EQS. "" THEN GOTO EXIT
$ EXCHANGE COPY /LOG CSA1:'SOURCE' *
$ EXIT:
$ DISMOUNT CSA1:
$ MOUNT/SYSTEM/FOREIGN/NOWRITE CSA1: "VAX console"
```

A VMS Disk Files and Volumes

A.1 Files–11 Disk Structure

The VMS system recognizes two disk file structures: Files–11 On–Disk Structure Level 1 and Files–11 On–Disk Structure Level 2. Files–11 On–Disk Structure Level 2 is the default disk structure of the VMS system, and Files–11 On–Disk Structure Level 1 is a structure used by DIGITAL's RSX–11M, RSX–11D, RSX–11M–PLUS, and IAS operating systems.

Nine files control the structure of a Files–11 On–Disk Structure Level 2 volume. Only five of these files are used for a Files–11 On–Disk Structure Level 1 volume. Table A–1 identifies all nine files, which are referred to as reserved files, and indicates to which Files–11 On–Disk Structure Level they pertain.

Table A–1 VMS Reserved Files

Reserved File	File Name	Structure Level 1	Structure Level 2
Index file	INDEXF.SYS;1	X	X
Storage bit map file	BITMAP.SYS;1	X	X
Bad block file	BADBLK.SYS;1	X	X
Master file directory	000000.DIR;1	X	X
Core image file	CORIMG.SYS;1	X	X
Volume set list file	VOLSET.SYS;1		X
Continuation file	CONTIN.SYS;1		X
Backup log file	BACKUP.SYS;1		X
Pending bad block	BADLOG.SYS;1		X

All the files listed in Table A–1 are listed in the master file directory (MFD), [000000].

A.1.1 Index File

Every Files–11 volume has an index file, which is created when the volume is initialized. This index file identifies the volume to the operating system as a Files–11 structure and contains the access data for all files on the volume. The index file, which is listed in the master file directory as INDEXF.SYS;1, contains the following information:

- Bootstrap block — The volume's bootstrap block is virtual block number 1 of the index file. If the volume is a system volume, this block contains a bootstrap program that loads the operating system into memory. If the volume is not a system volume, this block contains a program that displays the message that the volume is not the system device but a device that contains users' files only.

- Home block — The home block establishes the specific identity of the volume, providing such information as the volume name and protection, the maximum number of files allowed on the volume, and the volume ownership information. The home block is virtual block number 2 of the index file.
- Alternate home block — The alternate home block is a copy of the home block. It permits the volume to be used even if the primary home block is destroyed.
- Alternate index file header — The alternate index file header permits recovery of data on the volume if the primary index file header becomes damaged.
- Index file bit map — The index file bit map controls the allocation of file headers and thus the number of files on the volume. The bit map contains a bit for each file header allowed on the volume. If the value of a bit for a given file header is 0, a file can be created with this file header. If the value is 1, the file header is already in use.
- File headers — The largest part of the index file is made up of file headers. Each file on the volume has a file header, which describes such properties of the file as file ownership, creation date and time, file protection, and location of the data in the file. The file header contains all the information needed for gaining access to the file.

A.1.2 Storage Bit Map File

The storage bit map file controls the available space on a volume; this file is listed in the master file directory as BITMAP.SYS;1. It contains a storage control block, which consists of summary information intended to optimize the Files–11 space allocation, and the bit map itself, which lists the availability of individual blocks.

A.1.3 Bad Block File

The bad block file, which is listed in the master file directory as BADBLK.SYS;1, contains all the bad blocks on the volume. The system detects bad disk blocks dynamically and prevents their reuse once the files to which they are allocated have been deleted.

A.1.4 Master File Directory

The master file directory (MFD) itself is listed in the MFD as 000000.DIR;1. The MFD, which is the root of the volume's directory structure, lists the reserved files that control the volume structure and may list both users' files and users' file directories. Usually, however, the MFD is used to list the reserved files and users' file directories; users seldom enter files in the MFD, even on private volumes. In fact, on a private volume, it is most convenient for a user to create a directory that has the same name as the user's default directory on a system disk. For an explanation of users' file directories and file specifications, see the *VMS DCL Concepts Manual*.

When the VMS Backup Utility (BACKUP) creates sequential disk save sets, it stores the save set file in the MFD.

A.1.5 Core Image File

The core image file is listed in the MFD as CORIMG.SYS;1. It is not supported by the VMS operating system.

A.1.6 Volume Set List File

The volume set list file is listed in the MFD as VOLSET.SYS;1. This file is used only on relative volume 1 of a volume set. The file contains a list of the labels of all the volumes in the set and the name of the volume set.

A.1.7 Continuation File

The continuation file is listed in the MFD as CONTIN.SYS;1. This file is used as the extension file identifier when a file crosses from one volume to another volume of a loosely coupled volume set. This file is used for all but the first volume of a sequential disk save set.

A.1.8 Backup Log File

The backup log file is listed in the MFD as BACKUP.SYS;1. This file is reserved for future use.

A.1.9 Pending Bad Block Log File

The pending bad block log file is listed in the MFD as BADLOG.SYS;1. This file contains a list of suspected bad blocks on the volume that are not listed in the bad block file.

A.1.10 Files–11 On–Disk Structure Level 1 Versus Structure Level 2

For reasons of performance and reliability, Files–11 On–Disk Structure Level 2, a compatible superset of Structure Level 1, is the preferred disk structure on the VMS system. At volume initialization time (see the INITIALIZE command in the *VMS DCL Dictionary*), Structure Level 2 is the default. Structure Level 1 should be specified only for volumes that must be transportable to RSX–11M, RSX–11D, RSX–11M–PLUS, and IAS systems, as these systems support only that structure level. Additionally, you may be required to handle Structure Level 1 volumes transported to VMS from one of the above systems.

Where Structure Level 1 volumes are in use on the system, you should bear in mind the following limitations on them:

- Directories — No hierarchies of directories and subdirectories, and no ordering of directory entries (that is, the file names) in any way. RSX–11M, RSX–11D, RSX–11M–PLUS, and IAS systems do not support subdirectories and alphabetical directory entries.
- Disk quotas — Not supported.
- Multivolume files and volume sets — Not supported.
- Placement control — Not supported.

- Caches — No caching of file header blocks, file identification slots, or extent entries.
- System disk — Cannot be a Structure Level 1 volume.
- VAXcluster access — Local access only; cannot be shared across a cluster.
- Clustered allocation — Not supported.
- Backup home block — Not supported.
- Protection code E — Means extend for the RSX–11M operating system but is ignored by VMS.
- File versions — Limited to 32,767; version limits are not supported.
- Enhanced protection features (for example, Access Control Lists) — not supported.
- Long file names — Not supported.
- RMS journaling — Not supported.
- RMS execution statistics monitoring — Not supported.

Future enhancements to VMS software will be based primarily on Structure Level 2, so that further restrictions on Structure Level 1 volumes might be incurred.

B VMS ANSI-Labeled Magnetic Tape

This appendix briefly describes ANSI labels, data, and record formats supported by the VMS operating system. It also describes support for the various ANSI labels on VMS. Note, however, that the VMS software also supports the ISO standard. For a complete description of these labels, please refer to the ANSI X3.27–1978 or ISO 1001–1979 standard.

B.1 Logical Format of ANSI-Labeled Volumes

The format of VMS ANSI-labeled magnetic tape volumes is based on Level 3 of the ANSI standard for magnetic tape labels and file structure for information interchange. This standard specifies the format, content, and sequence of volume labels, file labels, and file structures. According to this standard, volumes are written and read on 9-track magnetic tape drives only. The contents of labels must conform to prescribed data and record formats. All labels must consist of ASCII "a" characters.

The VMS ANSI-labeled format allows you to write binary data in the file sections (see Figure B–1) of files. However, if you plan to use such files for information interchange across systems, make sure that the recipient system can read the binary data.

B.2 VMS Magnetic Tape Ancillary Control Process (MTAACP)

The VMS magnetic tape ancillary control process (MTAACP) is the internal software process of the operating system that interprets the logical format of VMS ANSI-labeled volumes. Transparent to your process, the MTAACP process reads, writes, and interprets VMS ANSI labels before passing this information to VMS Record Management Services (RMS) and Queue I/O ($QIO) system services. These services in turn read, write, and interpret the record format of the data written in the file section.

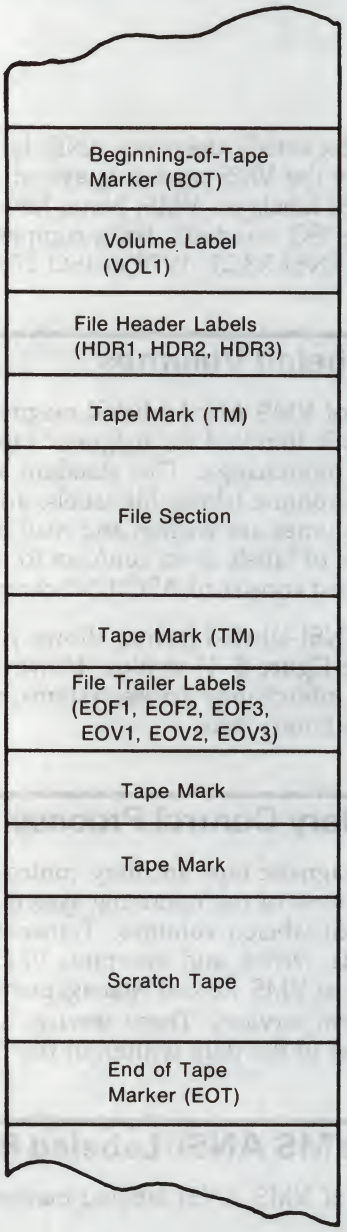
B.3 Basic Components of the VMS ANSI-Labeled Format

The format of VMS ANSI-labeled magnetic tape consists of four basic components:

- Beginning-of-tape (BOT) and end-of-tape (EOT) markers
- Tape marks
- File sections
- Volume, header, and trailer labels

Figure B–1 displays the arrangement and function of these components.

Figure B–1 Basic Layout of a VMS ANSI-Labeled Volume



B.3.1 Beginning-of-Tape and End-of-Tape Markers

Every volume has beginning-of-tape (BOT) and end-of-tape (EOT) markers. These markers are pieces of photoreflective tape that delimit the writable area on a volume. ANSI standards require that a minimum of 14 feet to a maximum of 18 feet of magnetic tape precede the BOT marker; a minimum of 25 feet to a maximum of 30 feet of magnetic tape, of which 10 feet must be writable, must follow the EOT marker. The EOT marker indicates the start of the end of the writable area of the tape, rather than the physical end of the tape. Therefore, data and labels can be written after the EOT marker.

B.3.2 Tape Marks

Tape marks separate the file labels from the file sections, separate one file from another, and denote the logical end-of-volume. On the basis of the number and relative placement of tape marks written on a volume, the VMS system determines whether a tape mark delimits a label, a file, or a volume.

Tape marks are written both singly and in pairs. Single tape marks separate either a file section from the header and trailer labels or one file from another. When written after a set of header labels, a single tape mark signals the beginning of a file section. When written before a set of trailer labels, a single tape mark indicates the end of a file section. When written after a trailer label set, a single tape mark separates one file from another.

Double tape marks indicate that either an empty file section exists or the logical end-of-volume has been reached. The VMS system creates an empty file when a volume is initialized.

B.3.3 Labels

Labels identify, describe, and control access to volumes and their files. The VMS ANSI-labeled format supports volume, header, and trailer labels. The volume labels are the first labels written on a volume. They identify the volume and the volume owner and specify access protection. Header and trailer labels are sets of labels that identify, describe, and delimit files. Header labels precede files; trailer labels follow files.

Table B–1 lists the labels supported by the VMS operating system. All other ANSI labels are ignored by VMS on input.

Although each type of label uses a different format to organize its contents, all labels conforming to Version 3 of the ANSI standard must consist of ASCII "a" characters. Some labels contain reserved fields designed for future system use or future ANSI standardization. Reserved fields also must consist of ASCII "a" characters; however, VMS ignores these characters on input.

B.4 Volume and File Configurations

VMS ANSI-labeled volumes support four file/volume configurations:

- Single-file/single-volume
- Single-file/multivolume
- Multifile/single-volume
- Multifile/multivolume.

All these configurations conform to the following guidelines:

- The file sequence number field allows as many as 9999 file sections for one file. In effect, the file length is unlimited.
- Only one file section of a given file is written on a volume.

- When multiple sections exist for one file, each file section is written to a separate volume in the volume set. The file section numbers of each section are written consecutively in ascending order (section *n+1* is written immediately following section *n*); file sections of other files are not interspersed.

Each of the file/volume configurations is illustrated in the subsections that follow.

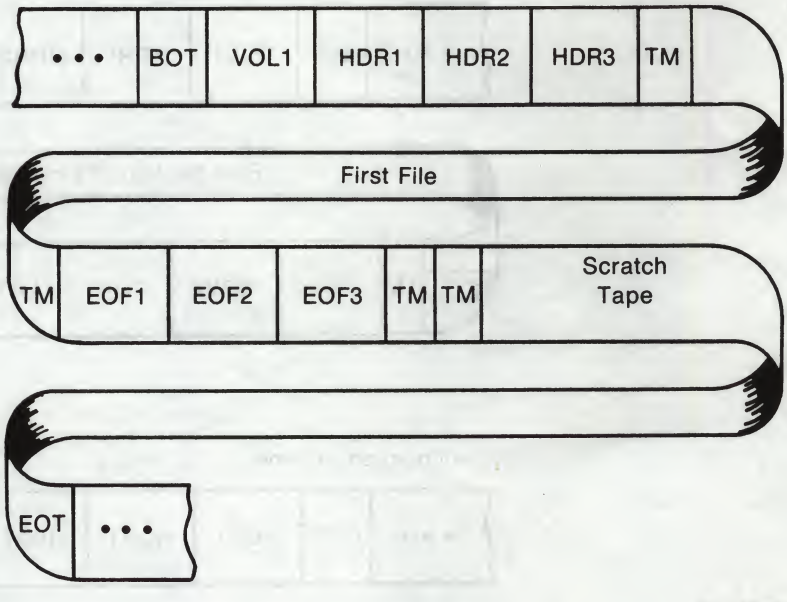
Table B–1 Labels and Components Supported by VMS

Symbol	Meaning
BOT	Beginning-of-tape marker
EOF1	First end-of-file label
EOF2	Second end-of-file label
EOF3	Third end-of-file label
EOF4	Fourth end-of-file label
EOT	End-of-tape marker label
EOV1	First end-of-volume label
EOV2	Second end-of-volume label
EOV3	Third end-of-volume label
EOV4	Fourth end-of-volume label
HDR1	First header label
HDR2	Second header label
HDR3	Third header label
HDR4	Fourth header label
VOL1	First volume label
VOL2	Second volume label
TM	Tape mark
TM TM	Double tape mark indicates an empty file section or the logical end-of-volume

B.4.1 Single-File/Single-Volume Configuration

A single-file/single-volume configuration consists of one file on one volume. The components of the ANSI-labeled format for this configuration are illustrated in Figure B–2.

Figure B–2 Single-File/Single-Volume Configuration



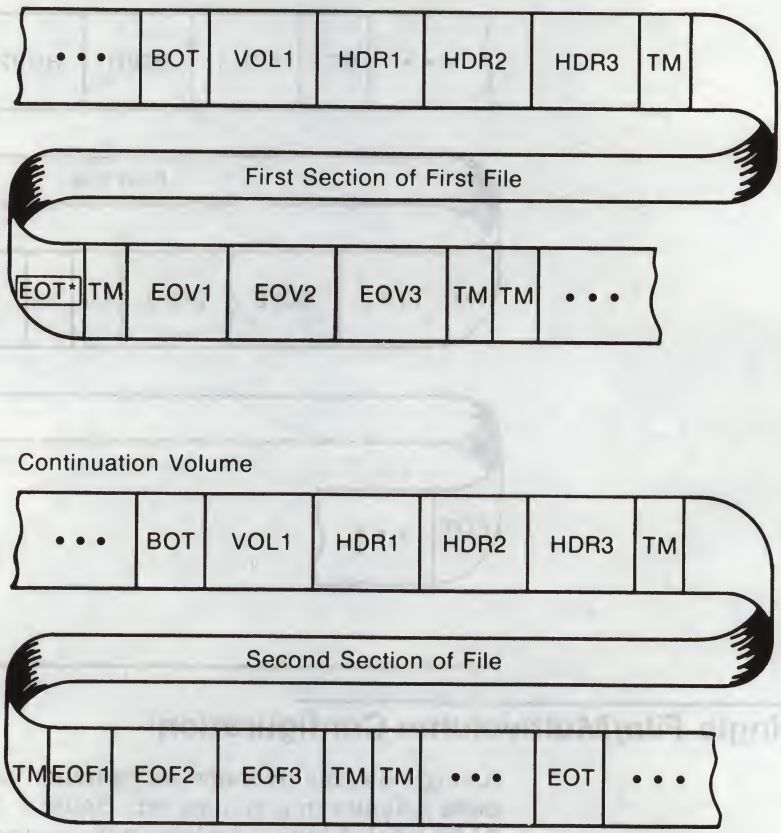
B.4.2 Single-File/Multivolume Configuration

A single-file/multivolume configuration consists of one file that spans two or more volumes in a volume set. Figure B–3 illustrates the components of the ANSI-labeled format for this configuration.

B.4.3 Multifile/Single-Volume Configuration

A multifile/single-volume configuration consists of two or more files on a single-volume. It is the most common file and volume configuration. Figure B–4 illustrates the components of the ANSI-labeled format for this configuration.

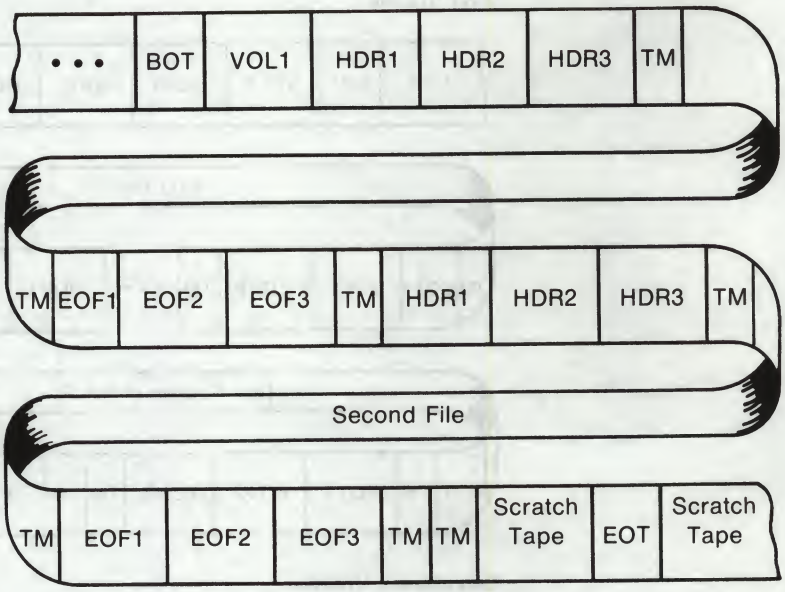
Figure B–3 Single-File/Multivolume Configuration



*When the driver encounters an EOT marker during a write operation, the MTAACP writes the appropriate trailer labels and performs a volume switch, if necessary.

ZK-347-81

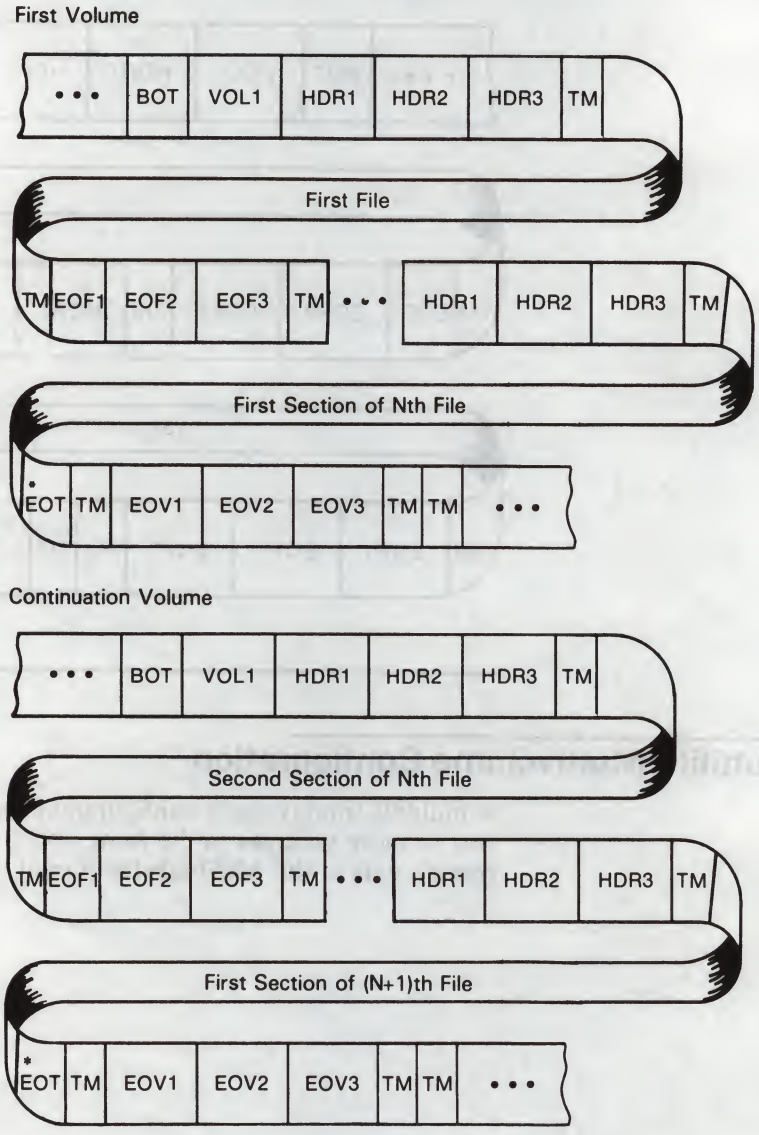
Figure B–4 Multifile/Single-Volume Configuration



B.4.4 Multifile/Multivolume Configuration

A multifile/multivolume configuration consists of two or more files that span two or more volumes in the same volume set. Figure B–5 illustrates the components of the ANSI-labeled format for this configuration.

Figure B–5 Multifile/Multivolume Configuration



*When the driver encounters an EOT marker during a write operation, the MTAACP writes the appropriate trailer labels and performs a volume switch, if necessary.

ZK-348-81

B.5 Volume Labels

The sections that follow describe the first volume (VOL1) and second volume (VOL2) labels.

B.5.1 VOL1 Label

The 80-character volume label (VOL1) is the first label written on a VMS ANSI-labeled volume. It defines the label type, name, and owner of the volume. Although there are many fields in a VOL1 label, this section describes only those fields that you can access or that can inhibit access to a volume and its files on VMS.

B.5.1.1 Volume Identifier Field

The volume identifier field is a 6-character field that contains the name of the volume. You specify the volume identifier in the command string when you initialize or mount a volume (see Chapter 3). The volume identifier consists of six ASCII "a" characters. Lowercase characters are not in the "a" set, but if you specify them, VMS will change them to uppercase. If you specify fewer than six characters, VMS pads the field by right-justifying the field with the ASCII space character.

B.5.1.2 Accessibility Field

The accessibility field is a one-character field that allows an installation to control access to a volume. See Section 2.1.3 for a description of accessibility support.

B.5.1.3 Implementation Identifier Field

The implementation identifier field contains the identifier of the implementation that creates the magnetic tape. This field controls how certain implementation-specific fields and volume labels are interpreted. The magnetic tape file system's implementation identifier is DECFILE11 A.

B.5.1.4 Owner Identifier Field

The owner identifier field is available to the user. This field does not affect the checking of a user's access to a volume, except as noted in Chapter 3.

B.5.2 VOL2 Label

In addition to the first volume (VOL1) label described above, VMS also provides a second volume (VOL2) label, the volume-owner field.

The volume-owner field contains the VMS protection information that has been written on the magnetic tape. A second volume label is written only if a VMS protection scheme had been specified on either the MOUNT or INITIALIZE command.

The volume-owner field also contains a value that incorporates the user identification code (UIC) with the VMS protection code specified for a volume. By default, VMS does not write a UIC to this field, thus allowing all users READ and WRITE access. Note, however, that EXECUTE and DELETE access are not applicable to magnetic tape volumes. Also note that, regardless of the protection code that you specify, both system users and the volume owner always have READ and WRITE access to a volume. The contents of the volume-owner field depends on the VMS protection code that you specify.

B.6 Header Labels

The VMS operating system supports four file-header labels: HDR1, HDR2, HDR3, and HDR4. The HDR3 and HDR4 labels are optional. The following sections describe and illustrate each file-header label.

B.6.1 HDR1 Label

Every file on a volume has a HDR1 label, which identifies and describes the file by supplying the VMS MTAACP with the following information:

- File identifier
- File-set identifier
- File section number
- File sequence number
- Generation and generation version numbers
- File creation and expiration dates
- Accessibility code
- Implementation identifier

B.6.1.1 File Identifier Field

The file identifier field contains the first 17 characters of the file name you specify. The remainder of the file name is written into the HDR4 label, provided that this label is allowed. If no HDR4 label is supported, a file name longer than 17 characters will be truncated. You may use either an ANSI file name or a VMS file specification of the following format:

```
filename.type;version
```

VMS file specifications are a subset of ANSI file names. However, ANSI file names are valid only for magnetic tape volumes; VMS file specifications are valid for disk and tape volumes. Both types of file specifiers are compatible with compatibility mode.

A VMS file specification consists of a file name, a file type, and an optional version number. Valid file names contain a maximum of 39 characters. Valid file types consist of a period followed by a maximum of 39 characters. The semicolon separates the version number from the file type.

Except for wildcard characters, only the characters A through Z, 0 through 9, and the special characters ampersand (&), hyphen (-), underscore (_) and dollar sign ($) are valid for VMS file names and types. The period and semicolon are the only other valid special characters, and they are always separators.

ANSI file names do not have a file type field. An ANSI file name consists of a 17-character name string, a period, a semicolon, and an optional version number. You can specify a name string consisting of a maximum of 17 ASCII "a" characters, but you must enclose the string in quotation marks (as in, for example, "file name"). When you specify fewer than 17 characters, the string is padded on the right with spaces to the 17-character maximum size. If you specify a file name that has trailing spaces, VMS truncates them when the

file name is returned. In addition, the space-padded field prevents you from specifying a unique file name that consists of spaces.

Although you can specify longer file names (up to 79 characters), only the first 17 characters of the file name will be used in interchange.

The quotation mark character requires special treatment because it is both the file name delimiter and a valid ASCII "a" character that can itself be embedded in the name string. You must specify two quotation marks for each one that you want the VMS system to interpret. The additional quotation mark informs VMS that one of the quotation marks is part of the name string, rather than a delimiter.

Embedded spaces also are valid characters, but embedded tabs are not. Lowercase characters are not in the ASCII "a" character set, but if you specify them, VMS converts them to uppercase characters.

If you do not specify a file type or version number on input, VMS will supply a period (the default file type) and a semicolon (the default version number). However, the period and semicolon will not be written to this field on the tape.

Although the VMS operating system considers version numbers for ANSI and VMS file names to be part of the file name specification, the version number of a file is not written to the file identifier field but is mapped to the generation number and generation version-number fields as described in Section B.6.1.4.

Examples below display ANSI file names. The input is the format that you specify. The output shown displays the VMS format returned to your process and the format written to the label. The number sign (#) in the examples indicates a space character. In the last example, a VMS file name is enclosed in quotation marks, like an ANSI file name, on input. However, the VMS operating system returns the file name to the process as a VMS file name, rather than an ANSI file name. Therefore, when you enclose a valid VMS file name in quotation marks on input, VMS parses the file name as a VMS file name.

```
     Input
"AB2&D""FgHI*k4""#-M";2
"##########"
"""""""""""""""""""""""""""";
"DWDEVOP.DAT"
"VMS_LONG_FILENAME.LONG_FILETYPE"

   Output to User Process
"AB2&D""FGHI*K4""#-M";2
"".;
"""""""""""""""""""""""""""".;
DWDEVOP.DAT;
VMS_LONG_FILENAME.LONG_FILETYPE

   Output to HDR1 Label
AB2&D"FGHI*K4"#-M
```

```
##################
"""""""""""""""####
DWDEVOP.DAT######
VMS_LONG_FILENAME
```

B.6.1.2 File-Set Identifier Field

The 6-character file-set identifier field denotes all files that belong to the same volume set. The file-set identifier for any file within a given volume set should always be the same as the file-set identifier of the first file on the first volume that you mount. For VMS, the file-set identifier is the same as the volume identifier of the first volume that you mount.

B.6.1.3 File Section Number and File Sequence Number Fields

The file section number is a 4-character field that specifies the number of the file section.

The file sequence number is a 4-character field that specifies the number of the file in a file set.

B.6.1.4 Generation Number and Generation Version-Number Fields

The generation number (a decimal number from 0001 to 9999) and generation version-number (a 2-digit decimal number) fields store the file version number specified on input and written by the system on output. The VMS operating system does not increment the version number of a file, even when the version of the specified file already exists on the volume. Therefore, if the file that you specify has the same file name and version number as an existing file, you will have at least two files with the same version number on the same volume set.

On input, VMS computes the version number by using this calculation:

```
version number =
[(generation number - 1) * 100] + generation version-number + 1
```

Version numbers larger than 32,767 are divided by 32,768; the integer remainder becomes the version number.

On output, the generation number is derived from the version number with this calculation:

```
generation number = [(version number - 1)/100] + 1
```

If there is a remainder after the version number is divided by 100, the remainder becomes the generation version number. It is not added to 1 to form the generation number.

B.6.1.5 Creation Date and Expiration Date Fields

The creation date field contains the date the file is created. The expiration date field contains the date the file expires. The system interprets the expiration date of the first file on a volume as the date that both the file and the volume expire. The creation and expiration dates are stored in the Julian format. This 6-character format (#YYDDD) consists of a space (#), the year, and the day. Only dates are relevant for these fields; time is always returned as 00:00:00:00.

By default, the current date is written to both the creation and expiration date fields when you create a file. Because the expiration date is the same as the creation date, the file expires on creation and you can overwrite it immediately. If the expiration date is a date that is later than the creation date and if the files you want to overwrite have not expired, you must specify the /OVERRIDE=EXPIRATION qualifier with the INITIALIZE or MOUNT command.

To write dates other than the VMS defaults in the date fields in this label, specify the creation date field (CDT) and the expiration date field (EDT) of the RMS date and time extended attribute block (XABDAT).

When you do not specify a creation date, VMS RMS defaults the current date to the creation date field. To specify a zero creation date, you must specify a year before 1900. If you specify a binary zero in the date field, the system will write the current date to the field.

For details on the XABDAT, see the *VMS Record Management Services Manual.*

B.6.1.6 Accessibility Field

The contents of this field are described in Section B.5.1.2.

B.6.1.7 Implementation Identifier Field

The implementation identifier field specifies, using ASCII "a" characters, an identification of the implementation that recorded the Volume Header Label Set.

B.6.2 HDR2 Label

The HDR2 label describes the record format, maximum record size, and maximum block size of a file.

B.6.2.1 Record Format Field

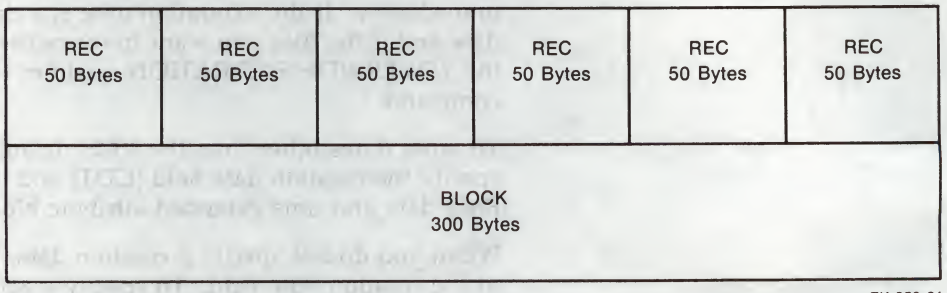
The record format field specifies the type of record format the file contains. The VMS operating system supports two record formats: fixed length (F) and variable length (D). When files contain record formats that the system does not support, it cannot interpret the formats and classifies them as undefined.

Fixed-length records are all the same length. No indication of the record length is required within the records because either the HDR2 label defines the record length or you specify the record length with the /RECORDSIZE qualifier. A fixed-length record can be a complete block, or several records can be grouped together in a block.

Fixed-length blocked records are padded to a multiple of 4 records. Variable-length records are padded to the block size. If a block is not filled, it will be padded with circumflex characters (^). The standard does not allow records containing only circumflexes; the system will interpret this as padding, not data.

Figure B–6 shows a block of fixed-length records. Each record has a fixed length of 50 bytes. All six records are contained in a 300-byte block. The records are blocked—that is, grouped together as one entity—to increase processing efficiency; when records are blocked, you can access many of them with one I/O request. The block size should be a multiple of the record size.

Figure B–6 Blocked Fixed-Length Records

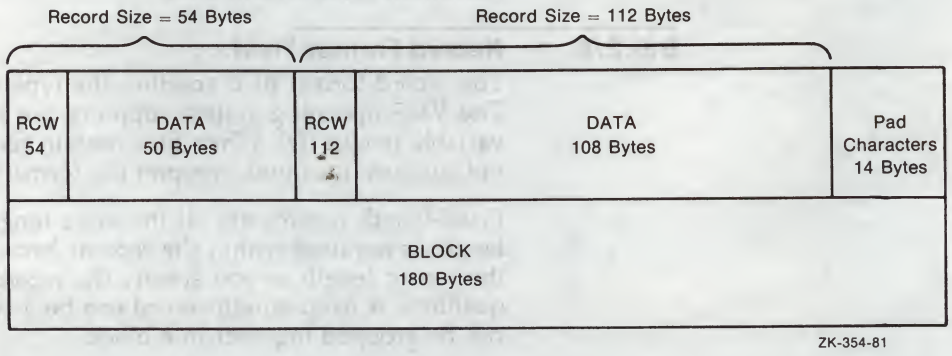


The size of a variable-length record is indicated by a record control word (RCW). The RCW consists of four bytes at the beginning of each record. A variable-length record can be a complete block, or several records can be grouped together in a block.

Two variable-length records are shown in Figure B–7. The first consists of 54 bytes, including the RCW. The second consists of 112 bytes, including the RCW. The records are contained in a 166-byte block.

Do not use system-dependent record formats on volumes used for information interchange. VMS system-dependent formats are stream and variable with fixed-length control (VFC).

Figure B–7 Variable-Length Records



B.6.2.2 Block Length Field

The block length field denotes the maximum size of the blocks. According to the ANSI standard, valid block sizes range from 18 to 2048 bytes. However, the VMS operating system allows you to specify a smaller or larger block size by using the /BLOCKSIZE qualifier with the MOUNT command. To specify the block size using VMS RMS, see the BLS field in the file access block (FAB) in the *VMS Record Management Services Manual*. When you specify a block size outside the ANSI standard range, the volume may not be processed correctly by other systems that support the ANSI standards.

B.6.2.3 Record Length Field

The record length field denotes either the size of fixed-length records or the maximum size of variable-length records in a file. Valid VMS record sizes vary, depending on the record format. The range for fixed-length records is 1 to 65,534 bytes; the range for variable-length records is 4 to 9,999 bytes, including the 4-byte RCW. Therefore, the maximum length of the data area of a variable-length record is 9,995 bytes. To comply with ANSI standards, the record size should not be larger than the maximum block size of 2,048 bytes, even though VMS allows larger record sizes (when the block size is larger).

For volumes containing files that do not have HDR2 labels, you must specify MOUNT/RECORDSIZE=n at mount time. The variable *n* denotes the record length in bytes. Files without HDR2 labels were created by a system that supports only level 1 or 2 of the ANSI standard for magnetic tape labels and file structure. Records should be fixed length because this is the only record format that either level supports. If you do not specify a record size, each block will be considered a single record.

B.6.2.4 Implementation-Dependent Field

The implementation-dependent field contains two 1-character subfields that describe how the VMS operating system interprets record format and form control.

The first subfield, character position 16, denotes whether the RMS attributes are in this label or the HDR3 label. If character position 16 contains a space, the RMS attributes are in the HDR3 label; if it contains any character other than a space, character position 16 is the first byte of the RMS attributes in the HDR2 label. The attributes appear in character positions 16 through 36 and 38 through 50.

For volumes created by any VMS version up to and including Version 2.0, the system-dependent fields of the HDR2 label contain the RMS file attributes in binary format. For volumes created on VMS versions later than Version 2.0 (Version 2.1 to the present), the magnetic tape file system stores the RMS attributes.

The second subfield, the form control field at character position 37, specifies the form control that defines the carriage control applied to records within a file. Possible values supported for VMS magnetic tape volumes are listed below.

A	First byte of record contains FORTRAN control characters.
M	The record contains all form control information.
space	Line-feed/carriage-return combination is to be inserted between records when the records are written to a carriage-control device, such as a line printer or terminal. If form control is not specified when a file is created, this is the default.

B.6.2.5 Buffer-Offset Length Field

For implementations that support buffer offsets, the buffer-offset length field indicates the length of information that prefixes each data block. The magnetic tape file system supports the input of buffer offset, which means that the buffer-offset length obtained from the HDR2 label (when reading the file) is used to determine the actual start of the data block. The magnetic tape file system does not support the writing of a buffer offset.

Note that, if you open a file for append or update access and the buffer-offset length is nonzero, the open operation will not succeed.

B.6.3 HDR3 Label

The HDR3 label describes the RMS file attributes. For RMS operations, data in the HDR3 label supersedes data in the HDR2 label.

Although the HDR3 label usually exists for every file on a VMS ANSI-labeled volume, there are two situations when this label will not be written. The first is when an empty dummy file is created during volume initialization; no HDR3 label is written because the empty file does not require RMS attributes. The second is when you specify MOUNT/NOHDR3 at mount time. You should use the /NOHDR3 qualifier when you create files on volumes that will be interchanged to systems that do not process HDR3 labels correctly.

The RMS attributes describe the record format of a file. These attributes are converted from 32 bytes of binary values to 64 bytes of ASCII representations of their hexadecimal equivalents for storage in the HDR3 label.

B.6.4 HDR4 Label

The HDR4 label contains the remainder of a VMS file name that would not fit in the HDR1 file identifier field.

B.7 Trailer Labels

The VMS operating system supports two sets of trailer labels: end-of-file (EOF) and end-of-volume (EOV). A trailer label is written for each header label.

EOF and EOV labels are identical to their file header label counterparts except that:

- The label identifier field (characters 1-3) contains EOF or EOV.
- The block count field (characters 55-60) in the EOF1 and EOV1 labels contains the number of data blocks in the file section.

The particular label that VMS writes depends on whether a file extends beyond a volume. If a file terminates within the limits of a volume, EOF labels are written to delimit the file (see Figure B–2). If a file extends across volume boundaries before terminating, EOV labels are written, indicating that the file continues on another volume (see Figure B–3).

Index

A

B

C

D

E

F

G

H

I

L

M

O

P

Q

R

S

T

U

V

W

X

Z

NOTES

NOTES

NOTES

NOTES

Reader's Comments

Guide to VMS Files and Devices
AA–LA06A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page Description

_____ _____

_____ _____

_____ _____

_____ _____

_____ _____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

— Do Not Tear - Fold Here and Tape ——————————————————

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

— Do Not Tear - Fold Here ——————————————————

Cut Along Dotted Line

Reader's Comments

Guide to VMS Files and Devices
AA–LA06A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

Do Not Tear - Fold Here and Tape

digital™

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

Do Not Tear - Fold Here

Cut Along Dotted Line